

USER GUIDE

Essential Studio

for EJ2 TypeScript

Version - v24.1.41 | Release Date - December 18, 2023

Gantt	50
Module in EJ2 JavaScript Gantt control	50
Data binding in EJ2 JavaScript Gantt control	50
Local data	50
Remote data.....	54
Split task.....	71
Limitations.....	74
Immutable in EJ2 JavaScript Gantt control	75
Limitations.....	76
Virtual scroll in EJ2 JavaScript Gantt control	76
Row virtualization	76
Timeline virtualization	78
Get filtered data when virtual scrolling is enabled.....	79
Limitations for virtual vcroll	81
Selection.....	81
Selection in EJ2 JavaScript Gantt control.....	81
Row selection in EJ2 JavaScript Gantt control	88
Cell selection in EJ2 JavaScript Gantt control	95
Filtering	99
Filtering in EJ2 JavaScript Gantt control.....	99
Filter menu in EJ2 JavaScript Gantt control	107
Excel like filter in EJ2 JavaScript Gantt control	110
Searching in EJ2 JavaScript Gantt control	111
Task dependency in EJ2 JavaScript Gantt control.....	118
Task relationship types	118
Define task relationship	118
Predecessor offset with duration units.....	120
Disabling automatic dependency offset updates	122
Validate predecessor links on editing	124
Dynamically show/hide the dependency line.....	128
Sorting in EJ2 JavaScript Gantt control	130
Sorting column on Gantt initialization	131
Sorting column dynamically.....	133
Clear all the sorted columns dynamically	134
Sorting events	135

Sorting Custom Columns.....	136
Baseline in EJ2 JavaScript Gantt control	138
Timeline.....	140
Timeline in EJ2 JavaScript Gantt control.....	140
Top tier and bottom tier in EJ2 JavaScript Gantt control.....	148
Zooming in EJ2 JavaScript Gantt control.....	153
Timezone in EJ2 JavaScript Gantt control	158
Understanding date manipulation in JavaScript.....	158
Display same time everywhere with no time difference.....	158
CRUD operations with timezone.....	160
Timezone methods	164
Event markers in EJ2 JavaScript Gantt control	165
Displaying eventMarkers in stacked manner.....	166
Label positions in EJ2 JavaScript gantt control	168
Managing event marker overlapping in EJ2 JavaScript gantt control.....	170
Work in EJ2 JavaScript Gantt control	171
Task type	173
Resources in EJ2 JavaScript Gantt control	175
Resource collection	175
Assign resource	176
Add/Edit resource collection	178
Resource view in EJ2 JavaScript Gantt control	179
Unassigned task	179
Resource task	179
Resource OverAllocation.....	181
Resource Multi Taskbar	183
Holidays in EJ2 JavaScript Gantt control	188
Tooltip in EJ2 JavaScript Gantt control	190
Enable tooltip.....	190
Timeline cells tooltip.....	192
Cell tooltip.....	193
Tooltip template	195
Appearance customization in EJ2 JavaScript Gantt control.....	200
Taskbar customization	200
Task labels	207

Connector lines	208
Customize rows and cells	210
Grid lines	212
Splitter.....	213
Duration unit in EJ2 JavaScript Gantt control	216
Duration units	216
Task scheduling in EJ2 JavaScript Gantt control	220
Automatically Scheduled Tasks.....	220
Manually Scheduled Tasks	221
Custom	223
Unscheduled Tasks.....	225
Define unscheduled tasks in data source	226
Working Time Range	228
Weekend/Non-working days	230
Critical path in EJ2 JavaScript Gantt control	231
Customize taskbar in critical path.....	232
Rows in EJ2 JavaScript Gantt control	234
Row height	234
Expand/Collapse Row	235
Customize rows.....	247
Styling alternate rows	249
Row spanning.....	250
Scrolling in EJ2 JavaScript Gantt control	251
Set width and height.....	251
Responsive with the parent container.....	253
Scroll To Date method	254
Set the vertical scroll position.....	255
Managing Tasks.....	257
Managing tasks in EJ2 JavaScript Gantt control.....	257
Adding new tasks in EJ2 JavaScript Gantt control	261
Editing tasks in EJ2 JavaScript Gantt control.....	264
Validation in EJ2 JavaScript Gantt control	273
Deleting tasks in EJ2 JavaScript Gantt control	277
Task bar editing in EJ2 JavaScript Gantt control	280
In dent and out dent in EJ2 JavaScript Gantt control	292

Splitting and merging tasks in EJ2 JavaScript Gantt control	294
Maintaining data in server in EJ2 JavaScript Gantt control	295
Columns	301
Columns in EJ2 JavaScript Gantt control.....	301
Column reordering in EJ2 JavaScript Gantt control	308
Column resizing in EJ2 JavaScript Gantt control	312
Column template in EJ2 JavaScript Gantt control.....	314
Column menu in EJ2 JavaScript Gantt control	317
Responsive columns in EJ2 JavaScript Gantt control	323
Check box columns in EJ2 JavaScript Gantt control.....	326
Column spanning in EJ2 JavaScript Gantt control.....	329
State persistence in EJ2 JavaScript Gantt control.....	332
Get or set localStorage value	332
Prevent columns from persisting.....	332
Persist the header template and header Text	334
Tool bar in EJ2 JavaScript Gantt control	336
Built-in toolbar items	336
Custom toolbar items	338
Built-in and custom items in toolbar	339
Enable/disable toolbar items	340
Add input elements to toolbar.....	342
Context menu in EJ2 JavaScript Gantt control.....	343
Custom context menu items.....	345
Excel Export.....	347
Excel export in EJ2 JavaScript Gantt control	347
Custom data source in EJ2 JavaScript Gantt control.....	348
Multiple gantt exporting in EJ2 JavaScript Gantt control	349
Pdf Export.....	363
Pdf export in EJ2 JavaScript Gantt control.....	363
Multiple gantt exporting in EJ2 JavaScript Gantt control	372
To customize PDF export	373
Customizing header and footer of PDF export in EJ2 JavaScript Gantt control.....	393
Data markers in EJ2 JavaScript Gantt control	400
Global local in EJ2 JavaScript Gantt control.....	402
Localization	402

Internationalization.....	406
Right to left (RTL)	409
See Also	412
Accessibility in EJ2 JavaScript Gantt control	412
WAI-ARIA attributes.....	413
Keyboard navigation	414
Ensuring accessibility	415
See also	415
Touch interaction in EJ2 JavaScript Gantt control	415
Tooltip	415
Context menu.....	415
Sorting.....	415
Column resize.....	416
Editing	417
Selection.....	419
Ej1 api migration in EJ2 JavaScript Gantt control	420
Members	422
Sorting.....	430
Filtering	431
Searching.....	432
Selection.....	432
Editing	433
Columns	436
Toolbar	436
ToolTip	437
Timeline.....	438
Rows.....	439
Resources	441
Baseline	442
Context Menu	442
Scheduling Tasks	442
Appearance and Customizations	443
Stripline	445
Holidays.....	446
Others	446

Methods.....	448
Events.....	450
Loading animation in EJ2 JavaScript Gantt control.....	452
Style and appearance in EJ2 JavaScript Gantt control.....	453
How To	454
Open add edit dialog in EJ2 JavaScript Gantt control	454
Change schedule date in EJ2 JavaScript Gantt control	456
Copy paste records in EJ2 JavaScript Gantt control.....	457
Maintain record index in EJ2 JavaScript Gantt control	459
Custom field in EJ2 JavaScript Gantt control	462
Maintain zoom to fit in EJ2 JavaScript Gantt control.....	465
Drag and drop in EJ2 JavaScript Gantt control.....	468
New row position in EJ2 JavaScript Gantt control	470
Render timeline from 1 to 365 days in EJ2 JavaScript Gantt control.....	473
Grid.....	474
Getting started in EJ2 JavaScript Grid control	474
Dependencies.....	474
Setup for local environment	475
Adding Syncfusion resources	475
Adding Grid control.....	478
Defining Row Data	479
Defining Columns	480
Module injection.....	480
Enable paging.....	481
Enable sorting	485
Enable filtering.....	489
Enable grouping	493
Run the application	497
Deploy the Application.....	501
See Also	501
Module in EJ2 JavaScript Grid control	501
Data Binding.....	502
Data binding in EJ2 JavaScript Grid control	502
Local data in EJ2 JavaScript Grid control.....	505
Remote data in EJ2 JavaScript Grid control	509

Immutable mode in EJ2 JavaScript Grid control	521
Limitations.....	525
Performance tips for EJ2 JavaScript Grid control.....	525
How to improve loading performance by binding large dataset.....	525
How to improve loading performance by binding large data by showing custom text or element.	526
How to improve loading performance by referring individual script and CSS	526
How to update cell values without frequent server calls	527
How to optimize server-side data operations with adaptors.....	527
How to avoid MaxJsonLength error while passing large amount of records	527
Microsoft excel limitation while exporting millions of records to excel file format.....	528
Columns	528
Columns in EJ2 JavaScript Grid control	528
Auto generated columns in EJ2 JavaScript Grid control	553
Headers in EJ2 JavaScript Grid control.....	559
Column template in EJ2 JavaScript Grid control.....	594
Complex data binding in EJ2 JavaScript Grid control.....	602
Foreign key column in EJ2 JavaScript Grid control	604
Auto fit columns in EJ2 JavaScript Grid control	618
Column reorder in EJ2 JavaScript Grid control	619
Column resizing in EJ2 JavaScript Grid control	628
Column chooser in EJ2 JavaScript Grid control.....	639
Column menu in EJ2 JavaScript Grid control	648
Column spanning in EJ2 JavaScript Grid control.....	660
Responsive columns in EJ2 JavaScript Grid control	667
Row	669
Row in EJ2 JavaScript Grid control	669
Row height in EJ2 JavaScript Grid control.....	678
Row template in EJ2 JavaScript Grid control	681
Detail template in EJ2 JavaScript Grid control.....	688
Row drag and drop in EJ2 JavaScript Grid control	698
Row spanning in EJ2 JavaScript Grid control	708
Cell	712
Cell in EJ2 JavaScript Grid control	712
Content in EJ2 JavaScript Grid control	718
Auto wrap in EJ2 JavaScript Grid control	719

Clip mode in EJ2 JavaScript Grid control	721
Editing	723
Edit in EJ2 JavaScript Grid control	723
Edit types in EJ2 JavaScript Grid control	736
In line editing in EJ2 JavaScript Grid control	762
Dialog editing in EJ2 JavaScript Grid control	779
Template editing in EJ2 JavaScript Grid control	791
Batch editing in EJ2 JavaScript Grid control	802
Command column editing in EJ2 JavaScript Grid control	814
Validation in EJ2 JavaScript Grid control	817
Persisting data in server in EJ2 JavaScript Grid control	825
Sorting in EJ2 JavaScript Grid control	831
Initial sort	833
Multi-column sorting	835
Sort order	837
Sort foreign key column based on Text	837
Sorting events	839
Custom sort comparer	841
Touch interaction	843
Grouping	843
Grouping in EJ2 JavaScript Grid control	843
Caption template in EJ2 JavaScript Grid control	856
Reordering on grouped columns in EJ2 JavaScript Grid control	861
Lazy load grouping in EJ2 JavaScript Grid control	862
Filtering	868
Filtering in EJ2 JavaScript Grid control	868
Filter bar in EJ2 JavaScript Grid control	875
Filter menu in EJ2 JavaScript Grid control	881
Excel like filter in EJ2 JavaScript Grid control	892
Searching in EJ2 JavaScript Grid control	897
Initial search	899
Search operators	900
Search by external button	901
Search specific columns	902
Search on each key stroke	905

Highlight the search text.....	907
Perform search operation in Grid using multiple keywords.....	909
See Also	912
Paging in EJ2 JavaScript Grid control	912
Template	914
Pager with Page Size Dropdown	916
How to render Pager at the Top of the Grid	918
See Also	920
Scrolling in EJ2 JavaScript Grid control	920
Set width and height.....	920
Responsive with parent container	922
Sticky Header	923
Scroll to selected row.....	925
Hide the scrollbar when the content is not overflown	927
Frozen in EJ2 JavaScript Grid control	928
Limitations of Frozen Grid.....	930
Freeze particular columns.....	930
Freeze Direction	932
Deprecated Methods	933
Virtual scroll in EJ2 JavaScript Grid control.....	935
Row Virtualization.....	935
Column Virtualization	937
Virtualization with Grouping.....	941
Limitations for virtual scrolling	941
Browser height limitation in virtual scrolling and solution	942
Infinite scroll in EJ2 JavaScript Grid control.....	948
InitialBlocks	951
Cache Mode	953
Limitations for Infinite Scrolling.....	956
Selection.....	956
Selection in EJ2 JavaScript Grid control	956
Row selection in EJ2 JavaScript Grid control	960
Cell selection in EJ2 JavaScript Grid control.....	970
Column selection in EJ2 JavaScript Grid control	974
Check box selection in EJ2 JavaScript Grid control	976

Aggregates	982
Aggregates in EJ2 JavaScript Grid control	982
Footer aggregate in EJ2 JavaScript Grid control	982
Group and caption aggregate in EJ2 JavaScript Grid control	988
Custom aggregate in EJ2 JavaScript Grid control	990
Reactive aggregate in EJ2 JavaScript Grid control	994
Print in EJ2 JavaScript Grid control	998
Page setup	1000
Print using an external button	1000
Print the visible page	1001
Print the hierarchy grid	1003
Print the master detail grid	1005
Print large number of columns	1007
Show or Hide columns while Printing	1007
Limitations of Printing Large Data	1009
See Also	1010
Adaptive in EJ2 JavaScript Grid control	1010
Render adaptive dialogs	1010
Vertical row rendering	1012
Hierarchy grid in EJ2 JavaScript Grid control	1015
ExpandAll by external button	1017
Expand child grid initially	1019
Dynamically load child grid data	1021
Bind hierarchy grid with different field	1023
Adding Record in ChildGrid	1025
Dynamically bind data to child grid based on parent row Data	1027
State Persistence	1030
State persistence in EJ2 JavaScript Grid control	1030
Get or set local storage value in EJ2 JavaScript Grid control	1035
Prevent to persist in EJ2 JavaScript Grid control	1035
Add to persist in EJ2 JavaScript Grid control	1037
Tool Bar	1039
Tool bar in EJ2 JavaScript Grid control	1039
Tool bar items in EJ2 JavaScript Grid control	1043
Custom tool bar in EJ2 JavaScript Grid control	1053

Pdf Export.....	1057
Pdf export in EJ2 JavaScript Grid control	1057
Export multiple grids in EJ2 JavaScript Grid control	1067
Pdf export options in EJ2 JavaScript Grid control	1071
Pdf cell style customization in EJ2 JavaScript Grid control	1092
Adding header and footer in EJ2 JavaScript Grid control	1097
Exporting hierarchy grid in EJ2 JavaScript Grid control	1104
Exporting grid with templates in EJ2 JavaScript Grid control	1106
Exporting grid in server in EJ2 JavaScript Grid control	1117
Excel Export.....	1124
Excel exporting in EJ2 JavaScript Grid control	1124
Export multiple grids in EJ2 JavaScript Grid control	1134
Excel export options in EJ2 JavaScript Grid control	1138
Excel cell style customization in EJ2 JavaScript Grid control	1157
Adding header and footer in EJ2 JavaScript Grid control	1164
Exporting hierarchy grid in EJ2 JavaScript Grid control	1167
Exporting grid with templates in EJ2 JavaScript Grid control	1169
Exporting grid in server in EJ2 JavaScript Grid control	1180
Global local in EJ2 JavaScript Grid control	1190
Localization	1190
Internationalization.....	1195
Right to left (RTL)	1198
See Also	1200
Accessibility in EJ2 JavaScript Grid control	1200
WAI-ARIA attributes.....	1200
Keyboard interaction	1201
Ensuring accessibility	1202
See also	1202
Clipboard in EJ2 JavaScript Grid control	1203
Copy to clipboard by external buttons	1204
AutoFill	1206
Paste.....	1208
Context menu in EJ2 JavaScript Grid control	1210
Custom context menu items.....	1213
Show context menu on left click.....	1215

Enable or disable context menu items	1218
Loading animation in EJ2 JavaScript Grid control	1221
Style And Appearance	1223
Style and appearance in EJ2 JavaScript Grid control	1223
Header in EJ2 JavaScript Grid control	1226
Paging in EJ2 JavaScript Grid control	1228
Sorting in EJ2 JavaScript Grid control.....	1232
Filtering in EJ2 JavaScript Grid control.....	1233
Grouping in EJ2 JavaScript Grid control.....	1241
Toolbar in EJ2 JavaScript Grid control.....	1245
Editing in EJ2 JavaScript Grid control.....	1246
Aggregate in EJ2 JavaScript Grid control.....	1251
Selection in EJ2 JavaScript Grid control	1252
Migration in EJ2 JavaScript Grid control	1253
Ej1 api migration in EJ2 JavaScript Grid control.....	1258
Sorting	1258
Grouping	1259
Filtering	1260
Searching.....	1261
Paging.....	1261
Selection.....	1262
Editing	1264
Resizing	1267
Reordering	1268
Context Menu	1268
Toolbar	1268
GridLines	1269
Templates.....	1270
Row/Column Drag and Drop.....	1270
Frozen Rows and Columns	1271
ForeignKey	1271
Auto Wrap.....	1271
Responsive	1272
State Persistence.....	1272
Right to Left - RTL.....	1272

ToolTip	1272
Aggregate/Summary	1272
Grid Export	1273
Columns	1273
Row	1275
Show/Hide Columns.....	1275
Column Chooser.....	1275
Header.....	1276
DataSource.....	1276
Print.....	1276
Scrolling.....	1276
PrimaryKey	1277
Grid.....	1277
How To	1279
Enable disable grid and its actions in EJ2 JavaScript Grid control	1279
Customize the edit dialog in EJ2 JavaScript Grid control	1282
Cascading drop down list with grid editing in EJ2 JavaScript Grid control	1284
Perform grid actions by keyboard short cut keys in EJ2 JavaScript Grid control.....	1287
Exporting grid in cordova application in EJ2 JavaScript Grid control.....	1289
Customize pager drop down in EJ2 JavaScript Grid control	1291
Hide the expand collapse icon in parent row in EJ2 JavaScript Grid control.....	1293
Select rows in grid based on certain condition in EJ2 JavaScript Grid control	1296
Collapse grouped rows at initial render in EJ2 JavaScript Grid control	1297
Merge duplicate cells in specific column and export in EJ2 JavaScript Grid control	1299
Get row cell index in EJ2 JavaScript Grid control	1303
Display null values at bottom in EJ2 JavaScript Grid control	1305
Enable editing in single click in EJ2 JavaScript Grid control	1307
Grid print in EJ2 JavaScript Grid control	1311
Change hierarchy grid icon in EJ2 JavaScript Grid control	1313
Render image in drop down list in EJ2 JavaScript Grid control	1315
Customize the Empty Record Template in EJ2 JavaScript Grid control	1317
HeatMap Chart.....	1319
Getting started in EJ2 JavaScript Heatmap chart control	1319
control Initialization	1319
Enable axis labels	1323

Add HeatMap title.....	1325
Enable legend.....	1326
Add data label.....	1327
Add custom cell palette	1329
Enable tooltip.....	1330
Working with data in EJ2 JavaScript Heatmap chart control.....	1332
Data adaptor	1332
Empty points	1340
Binding nested JSON data	1342
See Also	1344
Bubble heatmap in EJ2 JavaScript Heatmap chart control	1345
Bubble types	1345
Rendering mode in EJ2 JavaScript Heatmap chart control	1360
Axis in EJ2 JavaScript HeatMap chart control	1362
Types	1362
Inversed axis.....	1366
Opposed axis.....	1368
Axis labels customization	1370
Axis intervals	1378
Axis label increment.....	1380
Limiting labels in date-time axis.....	1381
Multilevel Labels	1384
Palette in EJ2 JavaScript Heatmap chart control	1387
Palette types	1387
Defining color stops	1390
Color range.....	1392
See Also	1393
Legend in EJ2 JavaScript Heatmap chart control	1394
Legend types	1395
Placement	1397
Alignment.....	1398
Legend dimensions	1400
Paging for legend	1401
Smart Legend	1403
Legend Selection	1405

Legend Title	1407
Appearance in EJ2 JavaScript HeatMap chart control	1408
Cell customization	1408
Background color	1412
Margin	1414
Title	1415
Data label	1416
See Also	1428
Dimensions in EJ2 JavaScript Heatmap chart control	1429
Size for container	1429
Size for heat map	1429
In Pixel	1429
In percentage	1430
Tooltip in EJ2 JavaScript Heatmap chart control	1431
Default Tooltip	1432
Tooltip template	1433
Customize the appearance of Tooltip	1434
Selection in EJ2 JavaScript Heatmap chart control	1436
Enable single cell selection	1438
Accessibility in EJ2 JavaScript HeatMap chart control	1441
Ensuring accessibility	1441
See also	1441
Events in EJ2 JavaScript HeatMap chart control	1441
cellClick	1441
cellDoubleClick	1443
cellRender	1444
cellSelected	1445
created	1447
legendRender	1448
load	1450
loaded	1451
resized	1452
tooltipRender	1454
How To	1455
Tooltip template in EJ2 JavaScript Heatmap chart control	1455

Legend customization in EJ2 JavaScript Heatmap chart control	1457
Ej1 api migration in EJ2 JavaScript Heatmap chart control	1459
Members	1459
Events.....	1463
Image Editor.....	1464
End-user capabilities in the EJ2 JavaScript Image Editor control	1464
Open an image	1464
Zooming	1464
Panning	1466
Cropping and image transformation.....	1466
Annotations.....	1467
Filtering and fine-tune	1468
Undo and redo the operations	1469
Reset an image.....	1470
Export an image	1471
Open and save in the EJ2 JavaScript Image Editor control	1472
Open an image	1472
Supported image formats	1474
Selection cropping in the EJ2 JavaScript Image Editor control	1478
Insert custom / square / circle region.....	1478
Insert selection based on aspect ratio	1480
Resize selections	1481
Crop an image	1482
Cropping event.....	1483
Annotation in the EJ2 JavaScript Image Editor control.....	1483
Text annotation.....	1484
Freehand drawing	1492
Shape annotation.....	1497
Delete a shape	1501
Image annotation.....	1502
Transform in the EJ2 JavaScript Image Editor control	1504
Rotate an image	1504
Flip an image	1506
Straighten an image	1508
Zoom in or out an image	1509

Maximum and Minimum zoom level	1513
Zooming event	1515
Rotating event.....	1515
Flipping event.....	1515
Toolbar in the EJ2 JavaScript Image Editor control.....	1516
Built-in toolbar items	1516
Add a custom toolbar item	1516
Show or hide a toolbar.....	1518
Show or hide a toolbar Item	1519
Enable or disable a toolbar item.....	1521
Enable or disable a contextual toolbar item.....	1521
Toolbar created event.....	1521
Toolbar item clicked event.....	1521
Toolbar template	1523
Customize Contextual Toolbar.....	1525
Add an additional contextual Toolbar item to text shape	1526
Localization in the EJ2 JavaScript Image Editor control.....	1528
Filters in the EJ2 JavaScript Image Editor control	1531
Apply filter effect	1531
Apply filter effect	1533
Finetuning in the EJ2 JavaScript Image Editor control.....	1534
Adjust the brightness, contrast, or sharpness	1534
Adjust the hue, exposure, blur, or opacity	1536
Finetune value changing event	1537
Frame in the EJ2 JavaScript Image Editor control.....	1538
Apply frame to the Image	1538
Frame changing event.....	1540
Resize	1540
Resizing event	1542
Quick access toolbar in the EJ2 JavaScript Image Editor control.....	1543
Add a custom toolbar item	1543
Undo and Redo in the EJ2 JavaScript Image Editor control.....	1544
Undo the action	1544
Redo the action.....	1545
Accessibility in EJ2 JavaScript Image Editor control.....	1546

Keyboard interaction	1547
Ensuring accessibility	1548
See also	1548
InPlaceEditor	1548
Controls in EJ2 JavaScript In place editor control	1548
Model configuration	1553
See Also	1554
Configuration in EJ2 JavaScript In place editor control	1554
Rendering modes	1554
Event actions for editing	1558
Action on focus out	1559
Display modes	1561
See Also	1562
Buttons in EJ2 JavaScript In place editor control	1562
See Also	1564
Server actions in EJ2 JavaScript In place editor control	1564
See Also	1566
Data binding in EJ2 JavaScript In place editor control	1567
Local	1567
Remote	1568
Integration in EJ2 JavaScript In place editor control	1570
As a string	1570
As a selector	1570
See Also	1572
Validation in EJ2 JavaScript In place editor control	1572
Validation Rules	1572
Accessibility in EJ2 JavaScript Inplace editor component	1572
Style in EJ2 JavaScript In place editor control	1572
Customizing the In-place Editor text	1572
Customizing the In-place Editor action buttons	1573
Localization in EJ2 JavaScript In place editor control	1574
Localization	1574
Right to left	1576
Format	1576
How To	1578

Dynamic edit mode in EJ2 JavaScript In place editor control	1578
Disable edit mode in EJ2 JavaScript In place editor control	1579
Custom indication in EJ2 JavaScript In place editor control	1580
Custom animation in EJ2 JavaScript In place editor control	1581
Kanban	1583
Getting started in EJ2 JavaScript Kanban control	1583
Dependencies.....	1583
Setup for local environment	1584
Adding Syncfusion resources	1584
Initialize the Kanban.....	1588
Populating cards.....	1590
Enable swimlane	1592
Columns in EJ2 JavaScript Kanban control.....	1593
Single-key mapping	1593
Multi-key mapping	1594
Header text	1596
Header template	1596
Toggle columns	1600
Stacked headers.....	1602
Cards in EJ2 JavaScript Kanban control.....	1604
Drag-and-drop.....	1604
Header.....	1604
Content	1606
Template	1606
Selection.....	1608
Data binding in EJ2 JavaScript Kanban control	1609
Local data	1609
Remote data.....	1611
Loading data via ajax.....	1623
Swimlane in EJ2 JavaScript Kanban control	1625
Render swimlane row	1625
Custom row text.....	1626
Template	1628
Sorting	1630
Drag-and-drop.....	1631

Create empty row	1633
Calculate cards count.....	1634
Enable frozen rows	1635
Drag and drop in EJ2 JavaScript Kanban control.....	1637
Internal drag and drop	1637
External drag and drop	1640
Sort in EJ2 JavaScript Kanban control	1648
Index.....	1648
DataSource Order	1652
Custom	1653
Change the direction.....	1655
Dialog in EJ2 JavaScript Kanban control.....	1656
Default Dialog.....	1656
Custom Fields.....	1658
Dialog Template	1663
Prevent Dialog.....	1666
Persisting data in server.....	1667
Tooltip in EJ2 JavaScript Kanban control	1673
Tooltip template	1673
Validation in EJ2 JavaScript Kanban control	1675
Minimum card limit.....	1675
Maximum card limit	1675
Virtualization in EJ2 JavaScript Kanban control	1677
Virtual scrolling	1677
Limitations for virtual scrolling	1680
Localization in EJ2 JavaScript Kanban control.....	1680
Loading translations.....	1681
Right to left (RTL)	1683
Dimensions in EJ2 JavaScript Kanban control.....	1685
Auto height and width	1685
Height and width in pixel	1686
Height and width in percentage	1688
Persistence in EJ2 JavaScript Kanban control	1689
Responsive mode in EJ2 JavaScript Kanban control	1690
Layouts	1690

Scrolling.....	1692
Selection.....	1693
Style in EJ2 JavaScript Kanban control	1696
To set fixed position to the Kanban header	1699
Accessibility in EJ2 JavaScript Kanban control	1699
WAI-ARIA attributes.....	1700
Keyboard interaction	1701
Disable keyboard interaction	1701
Ensuring accessibility	1703
See also	1703
How To	1703
Header double click in EJ2 JavaScript Kanban control.....	1703
Dynamically change columns in EJ2 JavaScript Kanban control	1704
Filter cards in EJ2 JavaScript Kanban control	1706
Search cards in EJ2 JavaScript Kanban control	1708
Ej1 api migration in EJ2 JavaScript Kanban control	1710
Columns	1710
Cards	1711
DataSource.....	1713
Stacked Headers.....	1716
WIP Validation.....	1716
Keyboard	1716
Toggle Columns.....	1717
Dialog Editing	1717
Dialog Editing Fields	1718
Tooltip	1719
Context Menu	1719
WorkFlows	1720
Filtering	1720
Searching.....	1720
External Drag And Drop	1721
Scrolling.....	1721
Card Selection and Hover.....	1722
Toolbar	1722
Responsive	1722

State Persistence.....	1722
Right to Left - RTL.....	1723
Linear Gauge	1723
Linear gauge dimensions in EJ2 JavaScript Linear gauge control	1723
Size for Linear Gauge	1723
Axis in EJ2 JavaScript Linear gauge control.....	1725
Setting the start value and end value of the axis.....	1725
Line Customization.....	1726
Ticks Customization	1727
Labels Customization	1729
Orientation.....	1734
Inverted Axis	1734
Opposed Axis.....	1735
Multiple Axes	1736
Ranges in EJ2 JavaScript Linear gauge control.....	1737
Customizing the range	1738
Setting the range color of the labels.....	1739
Multiple ranges	1740
Gradient Color.....	1741
Pointers in EJ2 JavaScript Linear gauge control.....	1744
Types of pointer	1745
Multiple pointers	1753
Pointer animation	1754
Gradient Color.....	1754
Annotations in EJ2 JavaScript Linear gauge control	1757
Adding annotation	1758
Customization	1758
Multiple annotations	1761
Animation in EJ2 JavaScript Linear Gauge control.....	1762
User interaction in EJ2 JavaScript Linear gauge control	1764
Tooltip	1764
Pointer Drag	1769
Linear gauge print and export in EJ2 JavaScript Linear gauge control	1770
Print.....	1770
Export.....	1771

Linear gauge appearance in EJ2 JavaScript Linear gauge control.....	1774
Customizing the Linear Gauge area	1774
Setting up the Linear Gauge title	1775
Customizing the Linear Gauge container	1776
Fitting the Linear Gauge to the control.....	1779
Accessibility in EJ2 JavaScript Linear Gauge control	1780
WAI-ARIA attributes.....	1780
Screen reading in Linear Gauge	1780
Ensuring accessibility	1780
See also	1781
Internationalization in EJ2 JavaScript Linear gauge control	1781
Numeric Format	1781
Events in EJ2 JavaScript Linear gauge control.....	1782
animationComplete	1782
annotationRender	1783
axisLabelRender	1784
beforePrint	1785
dragEnd	1785
dragMove	1786
dragStart	1787
gaugeMouseDown	1788
gaugeMouseLeave	1789
gaugeMouseMove	1790
gaugeMouseUp	1790
load	1791
loaded	1792
resized	1793
tooltipRender	1793
valueChange.....	1794
Methods in EJ2 JavaScript Linear gauge control.....	1795
setPointerValue.....	1795
setAnnotationValue	1796
refresh.....	1797
Ej1 api migration in EJ2 JavaScript Linear gauge control	1798
Linear gauge dimensions	1798

Line customizatn	1799
Ticks.....	1799
Labels	1801
Axis	1802
Ranges.....	1802
Bar Pointer	1803
Marker Pointer	1804
Annotation	1806
Tooltip	1807
Appearance of Linear Gauge.....	1808
Gauge Container type	1809
Events.....	1809
List Box	1811
Getting started in EJ2 JavaScript List box control	1811
Component Initialization.....	1811
Binding data source	1814
See Also	1815
Data binding in EJ2 JavaScript List box control	1815
Local Data.....	1816
Remote Data	1819
HTML element.....	1820
See Also	1823
Icons and templates in EJ2 JavaScript List box control	1823
Icons	1823
Templates.....	1824
See Also	1825
Selection in EJ2 JavaScript List box control	1826
Single selection	1826
Multiple selection	1827
See Also	1830
Sorting and grouping in EJ2 JavaScript List box control	1830
Sorting.....	1830
Grouping	1831
See Also	1833
Drag and drop in EJ2 JavaScript List box control	1834

Single listbox	1834
Multiple listbox	1835
See Also	1837
Dual list box in EJ2 JavaScript List box control	1837
Accessibility in EJ2 JavaScript List box control	1839
WAI-ARIA attributes	1840
Keyboard interaction	1840
Ensuring accessibility	1841
See also	1841
Style and appearance in EJ2 JavaScript List box control	1841
How To	1843
Enable scroller in EJ2 JavaScript List box control	1843
Add items in EJ2 JavaScript List box control	1844
Enable or disable items in EJ2 JavaScript List box control	1845
Form submit in EJ2 JavaScript List box control	1846
Select items in EJ2 JavaScript List box control	1848
ListView	1849
Getting started in EJ2 JavaScript Listview control	1849
control Initialization	1849
See Also	1852
Data binding in EJ2 JavaScript Listview control	1853
Bind to local data	1853
Bind to remote data	1856
Grouping in EJ2 JavaScript Listview control	1857
Customization	1859
Check list in EJ2 JavaScript Listview control	1859
Checkbox Position	1860
Nested list in EJ2 JavaScript Listview control	1861
Customizing templates in EJ2 JavaScript Listview control	1866
Header template	1866
Template	1868
Group template	1871
Virtualization in EJ2 JavaScript Listview control	1874
Module injection	1874
Getting started	1874

Conditional rendering	1877
Scrolling in EJ2 JavaScript Listview	1878
Accessibility in EJ2 JavaScript Listview component	1881
WAI-ARIA attributes	1882
Keyboard interaction	1883
Ensuring accessibility	1883
See also	1883
Style in EJ2 JavaScript Listview control	1883
Customizing ListView	1883
Customizing the list items	1884
Customizing ListView's header	1884
Customizing group header of ListView	1884
Customizing the hover state of ListView control	1884
Customizing selected item of ListView control	1885
Ej1 api migration in EJ2 JavaScript Listview control	1885
How To	1887
Get selected items from listview in EJ2 JavaScript Listview control	1887
Add and remove list items from listview in EJ2 JavaScript Listview control	1889
Dynamic templates in listview based on device in EJ2 JavaScript Listview control	1891
Create mobile contact layout from listview in EJ2 JavaScript Listview control	1894
Filter and search list items using listview in EJ2 JavaScript Listview control	1897
Listview with hyper link navigation in EJ2 JavaScript Listview control	1899
Chat window user interface using listview in EJ2 JavaScript Listview control	1900
Drag and drop list items in listview in EJ2 JavaScript Listview control	1904
Create dual list from listview in EJ2 JavaScript Listview control	1906
Dynamic listview by loading ajax html content in EJ2 JavaScript Listview control	1912
Display spinner for list item loading from remote data in EJ2 JavaScript Listview control	1913
Dynamic hierarchical listview in EJ2 JavaScript Listview control	1916
Hide checkbox in listview in EJ2 JavaScript Listview control	1918
Display items count in group header in EJ2 JavaScript Listview control	1925
Customize listview as grid layout in EJ2 JavaScript Listview control	1926
default-list .e-list-item {	1927
Customize listview with dynamic tags in EJ2 JavaScript Listview control	1932
Fetch selected items from listview template sample in EJ2 JavaScript Listview control	1938
Trace events of listview in EJ2 JavaScript Listview control	1940

Customize nested listview as bread crumbs in EJ2 JavaScript Listview control	1942
Integrate pager component with listview in EJ2 JavaScript Listview control	1949
Maps	1951
Populate data in EJ2 JavaScript Maps control	1951
Geometry types.....	1951
Shape data	1951
Data source	1951
Data binding.....	1953
Binding complex data source	1955
Layers in EJ2 JavaScript Maps control.....	1958
Multilayer	1958
SubLayer.....	1958
Displaying layer in the view	1960
See Also	1961
Providers	1961
Map provider in EJ2 JavaScript Maps control.....	1961
Bing maps in EJ2 JavaScript Maps control	1966
Azure maps in EJ2 JavaScript Maps control	1972
Other maps in EJ2 JavaScript Maps control	1976
Customization in EJ2 JavaScript Maps control.....	1985
Setting the size for Maps	1985
Maps title	1986
Setting theme.....	1987
Customizing Maps container	1989
Customizing Maps area.....	1990
Customizing the shapes	1991
Setting color to the shapes from the data source	1992
Applying border to individual shapes	1993
Projection type.....	1995
Color mapping in EJ2 JavaScript Maps control	1997
Range color mapping	1997
Equal color mapping	1999
Desaturation color mapping	2001
Multiple colors for a single shape	2002
Color for items excluded from color mapping	2004

Color mapping for bubbles	2005
Data label in EJ2 JavaScript Maps control.....	2006
Adding data labels.....	2006
Customization	2008
Label animation.....	2010
Smart labels.....	2011
Intersect action	2012
Adding data label as a template	2013
Polygon shape in EJ2 JavaScript Maps control	2014
Markers in EJ2 JavaScript Maps control.....	2017
Adding marker.....	2017
Adding marker template	2019
Customization	2020
Marker shapes	2021
Rendering Marker shape as image	2022
Multiple marker groups	2023
Customize marker shapes from data source	2024
Repositioning the marker using drag and drop	2027
Marker zooming.....	2030
Marker clustering.....	2032
Customization of marker cluster.....	2033
Marker cluster expand.....	2035
Tooltip for marker	2036
See Also	2038
Bubble in EJ2 JavaScript Maps control.....	2038
Bubble shapes	2039
Customization	2040
Setting color to the bubbles from the data source.....	2041
Bubble sizing	2042
Multiple bubble groups.....	2043
Enable tooltip for bubble	2045
Legend in EJ2 JavaScript Maps control	2046
Positioning of the legend	2046
Legend mode.....	2048
Legend for shapes	2050

Legend Shape	2052
Customization	2052
Legend for items excluded from color mapping	2054
Hide desired legend items	2056
Hide legend items based on data source value	2057
Bind legend item text from data source	2058
Hide duplicate legend items	2059
Toggle option in legend	2060
Enable legend for bubble	2062
Enable legend for markers	2063
Annotations in EJ2 JavaScript Maps control	2065
Annotation	2065
Annotation customization	2066
Multiple Annotation.....	2069
Navigation line in EJ2 JavaScript Maps control.....	2070
Customization	2070
Enabling the arrows	2071
User interactions in EJ2 JavaScript Maps control	2072
Zooming	2073
Selection.....	2081
Highlight	2091
Tooltip	2097
See Also	2102
Print in EJ2 JavaScript Maps control	2103
Print.....	2103
Export.....	2104
Accessibility in EJ2 JavaScript Maps control	2109
WAI-ARIA attributes.....	2109
Screen reading in Maps.....	2109
Keyboard Navigation.....	2110
Ensuring accessibility	2110
See also	2110
Internationalization in EJ2 JavaScript Maps control	2110
Globalization	2110
Numeric Format	2111

Localization in EJ2 JavaScript Maps control.....	2112
Methods in EJ2 JavaScript Maps control	2113
Methods.....	2113
getMinMaxLatitudeLongitude	2114
Ej1 api migration in EJ2 JavaScript Maps control.....	2115
Size Customization	2115
Title and Subtitle Customization.....	2116
Layer Customization.....	2116
Shape Customization	2117
Marker Customization	2118
Bubble Customization	2119
DataLabel Customization	2121
Legend Customization.....	2122
Zooming Customization	2124
Highlight And Selection Customization.....	2126
Navigation Line Customization	2127
Tooltip Customization	2128
Annotation Cutomization.....	2128
Maps Other Properties Customization	2129
Events.....	2130
How To	2132
Annotation in EJ2 JavaScript Maps control.....	2132
annotation {	2135
Custom path in EJ2 JavaScript Maps control	2136
Drilldown in EJ2 JavaScript Maps control	2137
Marker type in EJ2 JavaScript Maps control.....	2139
Multiple layer in EJ2 JavaScript Maps control	2142
Navigation line in EJ2 JavaScript Maps control.....	2144
Zooming in EJ2 JavaScript Maps control	2148
MaskedTextBox.....	2149
Mask configuration in EJ2 JavaScript Maskedtextbox control.....	2149
Standard mask elements.....	2149
Custom mask elements.....	2152
Prompt character	2155
Accessibility in EJ2 JavaScript Maskedtextbox control	2156

WAI-ARIA attributes.....	2157
Ensuring accessibility	2158
See also	2158
Style appearance in EJ2 JavaScript Maskedtextbox control	2158
Customizing the appearance of MaskedTextBox wrapper element.....	2158
Customizing the MaskedTextBox element on hovering	2158
How To	2159
Perform custom validation using form validator in EJ2 JavaScript Maskedtextbox control	2159
Set cursor position while focus on the input textbox in EJ2 JavaScript Maskedtextbox control ...	2161
Display numeric keypad when focus on mobile devices in EJ2 JavaScript Maskedtextbox control	2164
Customize the ui appearance of the control in EJ2 JavaScript Maskedtextbox control.....	2165
Ej1 api migration in EJ2 JavaScript Maskedtextbox control	2166
Common.....	2167
Mask Configuration.....	2169
Validation	2170
Mention.....	2171
Working with data in EJ2 JavaScript Mention control	2171
Binding local data.....	2171
Binding remote data	2175
See also	2178
Mention integration in EJ2 JavaScript Rich text editor control	2178
See Also	2181
Filtering data in EJ2 JavaScript Mention control.....	2181
Limit the minimum filter character.....	2181
Change the filter type	2182
Allow spacing between search.....	2183
Customize the suggestion item count	2184
See also	2186
Sorting in EJ2 JavaScript Mention control	2186
Templates in EJ2 JavaScript Mention control	2187
Item template	2187
Display template	2188
No records template	2190
Spinner template	2191

See also	2192
Localization in EJ2 JavaScript Mention control	2192
Loading translations	2192
See also	2194
Customization in EJ2 JavaScript Mention control	2194
Show or hide mention character	2194
Adding the suffix character after selection	2195
Configure the popup list	2196
Trigger character	2198
Accessibility in EJ2 JavaScript Mention control	2198
WAI-ARIA attributes	2199
Keyboard interaction	2199
Ensuring accessibility	2200
See also	2200
Menu Bar	2201
Icons and sub menu items in EJ2 JavaScript Menu control	2201
Icons	2201
Navigation	2202
Multilevel nesting	2204
See Also	2207
Data source binding and custom menu items in EJ2 JavaScript Menu control	2207
Data binding	2207
Custom menu items	2213
See Also	2215
Use case scenarios in EJ2 JavaScript Menu control	2216
Scrollable menu	2216
Menu in toolbar	2219
Hamburger menu	2221
Mobile view	2225
Accessibility in EJ2 JavaScript Menu control	2228
WAI-ARIA attributes	2229
Keyboard interaction	2229
Ensuring accessibility	2230
See also	2230
Style and appearance in EJ2 JavaScript Menu control	2230

How To	2230
Change orientation in EJ2 JavaScript Menu control	2230
Customize menu using css in EJ2 JavaScript Menu control	2232
Customize menu using events in EJ2 JavaScript Menu control	2233
Customize menu items in EJ2 JavaScript Menu control	2235
Create mnemonic ui in menuitem in EJ2 JavaScript Menu control	2241
Change sub menu position in EJ2 JavaScript Menu control	2243
Rounded corner in EJ2 JavaScript Menu control	2245
Change animation settings in EJ2 JavaScript Menu control	2247
Right to left in EJ2 JavaScript Menu control	2249
Set title in EJ2 JavaScript Menu control.....	2251
Menu item click in EJ2 JavaScript Menu control.....	2252
Ej1 api migration in EJ2 JavaScript Menu control.....	2254
Properties.....	2254
Methods.....	2256
Events.....	2257
Message	2258
Severities in EJ2 JavaScript Message control	2258
Variants in EJ2 JavaScript Message control	2260
Icons in EJ2 JavaScript Message control	2263
No Icon	2263
Custom Icon	2265
Close Icon	2267
Customization in EJ2 JavaScript Message control	2269
Content Alignment.....	2269
Rounded and Square.....	2271
CSS Message.....	2272
Template in EJ2 JavaScript Message control	2275
Accessibility in EJ2 JavaScript Message control	2276
WAI-ARIA attributes.....	2277
Keyboard interaction	2278
Ensuring accessibility	2278
See also	2278
MultiSelect.....	2278
Tags in EJ2 JavaScript Multi select control.....	2278

Select element	2278
UL element	2279
Input element	2280
Data binding in EJ2 JavaScript Multi select control	2281
Binding local data	2281
Binding remote data	2285
See Also	2286
Templates in EJ2 JavaScript Multi select control	2286
Item template	2286
Value template	2287
Group template	2289
Header template	2290
Footer template	2292
No records template	2293
Action failure template	2294
See Also	2295
Grouping in EJ2 JavaScript Multi select control	2295
HTML select	2297
Customization	2298
Grouping with CheckBox	2298
See Also	2300
Filtering in EJ2 JavaScript Multi select control	2300
Limit the minimum filter character	2301
Change the filter type	2303
Case sensitive filtering	2304
Diacritics Filtering	2306
See Also	2307
Custom value in EJ2 JavaScript Multi select control	2307
Checkbox in EJ2 JavaScript Multi select control	2309
Select All	2310
Selection Limit	2312
Selection Reordering	2313
See Also	2315
Chip customization in EJ2 JavaScript Multi select control	2315
Localization in EJ2 JavaScript Multi select control	2316

Loading translations	2316
See Also	2318
Style in EJ2 JavaScript Multi select control	2318
Customizing the background color of wrapper element	2318
Customizing the appearance of the delimiter wrapper element	2318
Customizing the appearance of chips	2319
Customizing the dropdown icon's color	2319
Customizing the focus color	2319
Customizing the disabled component's text color	2319
Customizing the color of the placeholder text	2320
Customizing the placeholder to add mandatory indicator(*).....	2320
Customizing the float label element's focusing color	2320
Customizing the outline theme's focus color	2320
Customizing the background color of focus, hover, and active item's	2321
Customizing the appearance of pop-up element	2321
Customizing the color of the checkbox.....	2321
Accessibility in EJ2 JavaScript Multi select control	2322
WAI-ARIA attributes.....	2323
Keyboard interaction	2323
Ensuring accessibility	2325
See also	2325
How To	2325
Icons support in EJ2 JavaScript Multi select control	2325
Achieve virtual scrolling in EJ2 JavaScript Multi select control.....	2326
Validation in EJ2 JavaScript Multi select control	2328
NumericTextBox.....	2330
Formats in EJ2 JavaScript Numerictextbox control	2330
Standard formats	2331
Custom formats	2332
Globalization in EJ2 JavaScript Numerictextbox control	2334
Localization	2334
Internationalization.....	2336
Right to Left(RTL)	2339
Accessibility in EJ2 JavaScript Numerictextbox control	2340
WAI-ARIA attributes.....	2341

Keyboard interaction	2342
Ensuring accessibility	2343
See also	2343
Style appearance in EJ2 JavaScript Numerictextbox control	2344
Customizing the appearance of NumericTextBox wrapper element.....	2344
Customizing the NumericTextBox icons	2344
How To	2344
Customize the ui appearance of the control in EJ2 JavaScript Numerictextbox control.....	2344
Customize the spin buttons up and down arrow in EJ2 JavaScript Numerictextbox control	2346
Customize the step value and hide spin buttons in EJ2 JavaScript Numerictextbox control	2348
Maintain trailing zeros in numerictextbox in EJ2 JavaScript Numerictextbox control	2349
Perform custom validation using form validator in EJ2 JavaScript Numerictextbox control	2351
Prevent nullable input in numerictextbox in EJ2 JavaScript Numerictextbox control.....	2353
Ej1 api migration in EJ2 JavaScript Numerictextbox control	2355
Common.....	2355
Globalization	2357
Group	2357
Numeric configuration	2357
Number Formats	2359
Validation	2359
Pager	2360
Getting started in EJ2 JavaScript Pager control	2360
Dependencies.....	2360
Setup for local environment	2360
Adding Syncfusion resources	2360
Adding Pager control	2362
Page Size	2363
Page Count	2364
Run the application	2366
Style and appearance in EJ2 JavaScript Pager control	2367
Customizing the Pager	2367
Accessibility in EJ2 JavaScript Pager control	2368
WAI-ARIA.....	2368
Keyboard navigation	2369
PDF Viewer.....	2369

Getting Started.....	2369
Getting started in Standalone PDF Viewer control.....	2369
Getting started in EJ2 JavaScript PDF Viewer control.....	2371
Feature module in EJ2 JavaScript Pdfviewer control.....	2374
See also	2375
Open PDF files.....	2375
Opening a PDF from URL.....	2375
Opening a PDF from base64 data	2378
Opening a PDF from database	2378
Saving PDF file.....	2381
Save PDF file to Server	2381
Download PDF file as a copy	2382
Save PDF file to Database	2383
Toolbar in EJ2 JavaScript Pdfviewer control	2385
Show/Hide the default toolbar	2386
Show/Hide the default toolbaritem.....	2389
See also	2392
Navigation in EJ2 JavaScript Pdfviewer control	2392
Toolbar page navigation option.....	2392
Bookmark navigation	2396
Thumbnail navigation	2398
Hyperlink navigation	2399
Table of content navigation	2399
See also	2400
Magnification in EJ2 JavaScript Pdfviewer control	2400
See also	2402
Accessibility in Syncfusion PDF Viewer components.....	2403
WAI-ARIA attributes.....	2404
Keyboard interaction	2404
Ensuring accessibility	2407
See also	2407
Text search in EJ2 JavaScript Pdfviewer control	2407
See also	2409
Annotation	2409
Interaction mode in EJ2 JavaScript Pdfviewer control.....	2409

Selection mode	2409
Panning Mode	2411
See also	2412
Form Designer	2412
Create programmatically in EJ2 JavaScript Pdfviewer control	2412
Create with user interface interaction in EJ2 JavaScript Pdfviewer control	2447
Print in EJ2 JavaScript Pdfviewer control	2452
See also	2455
Download in EJ2 JavaScript Pdfviewer control	2455
How to get the base64 string while downloading the PDF document	2457
See also	2458
Globalization in EJ2 JavaScript Pdfviewer control	2458
Server Deployment	2464
Pdfviewer server docker image overview in EJ2 JavaScript Pdfviewer control	2464
Provide your license key for activation	2465
Provide your license key for activation	2467
How to deploy docker image in azure app service for container in EJ2 JavaScript Pdfviewer control	2467
How to deploy docker image in azure kubernetes service in EJ2 JavaScript Pdfviewer control	2469
How to deploy pdfviewer server app in azure app service from visual studio in EJ2 JavaScript Pdfviewer control	2471
How To	2474
Create pdfviewer service in EJ2 JavaScript Pdfviewer control	2474
Create pdfviewer service in EJ2 JavaScript Pdfviewer control	2484
Change author name using annotation settings in EJ2 JavaScript Pdfviewer control	2500
Customization in EJ2 JavaScript Pdfviewer control	2501
magnificationToolbar {	2504
magnificationToolbar.e-toolbar .e-tbar-btn {	2504
topToolbar {	2505
popup .e-dlg-content {	2505
Highlight underline strikeout text in EJ2 JavaScript Pdfviewer control	2522
Enable resize in EJ2 JavaScript Pdfviewer control	2523
Customize text search color in EJ2 JavaScript Pdfviewer control	2523
Disable context menu in EJ2 JavaScript Pdfviewer control	2524
Extract texts in EJ2 JavaScript Pdfviewer control	2525

Import annotations in EJ2 JavaScript Pdfviewer control	2525
Identify added annotation mode in EJ2 JavaScript Pdfviewer control	2529
Overlapped annotation in EJ2 JavaScript Pdfviewer control	2529
Display custom tool tip for annotation in EJ2 JavaScript Pdfviewer control	2529
Select multi page annotations in EJ2 JavaScript Pdfviewer control	2530
Disable tile rendering in EJ2 JavaScript Pdfviewer control	2531
Add header value in EJ2 JavaScript Pdfviewer control	2531
Print document in EJ2 JavaScript Pdfviewer control	2532
Unload document in EJ2 JavaScript Pdfviewer control.....	2532
Load document in EJ2 JavaScript Pdfviewer control	2533
Import export annotation object in EJ2 JavaScript Pdfviewer control	2533
Delete annotation in EJ2 JavaScript Pdfviewer control	2534
Open thumbnail in EJ2 JavaScript Pdfviewer control	2535
Annotation selectors in EJ2 JavaScript Pdfviewer control	2535
Clear annotations in EJ2 JavaScript Pdfviewer control	2536
Resolve unable to find an entry point error in EJ2 JavaScript Pdfviewer control.....	2536
Customize toolbar in PDF Viewer component.....	2537
Troubleshooting.....	2541
Document Loading Issues in Version 23.1 or Newer	2541
Pivot Table	2542
Getting started in EJ2 JavaScript Pivotview control.....	2542
Dependencies.....	2542
Setup for local environment	2542
Adding Syncfusion resources	2543
Browser compatibility	2547
Initializing pivot table component in the application	2547
Assigning sample data to pivot table component	2549
Adding fields to row, column, values and filters axes.....	2550
Apply formatting to value fields	2551
Enable Grouping Bar	2552
Enable Pivot Field List	2553
Calculated field	2555
Exploring Filter axis	2556
Run the application	2558
Deploy the Application.....	2559

Data binding in EJ2 JavaScript Pivotview control.....	2560
JSON	2560
CSV	2565
Remote Data Binding	2569
Mapping	2576
Values in row axis.....	2579
Values at different positions	2581
Show 'no data' items.....	2582
Always shows the value headers	2584
Customize empty value cells.....	2585
Events.....	2586
See Also	2591
Connecting To Data Source.....	2591
MySQL in EJ2 JavaScript Pivotview control.....	2591
Microsoft SQL server in EJ2 JavaScript Pivotview control	2596
PostgreSQL in EJ2 JavaScript Pivotview control.....	2601
Oracle in EJ2 JavaScript Pivotview control.....	2606
MongoDB in EJ2 JavaScript Pivotview control	2611
Elasticsearch in EJ2 JavaScript Pivotview control	2617
Snowflake in EJ2 JavaScript Pivotview control.....	2621
Olap in EJ2 JavaScript Pivotview control.....	2627
Getting Started.....	2627
Data Binding.....	2668
OLAP Cube: Elements.....	2677
Server side pivot engine in EJ2 JavaScript Pivotview control	2680
Quick steps to render the Pivot Table by using the server-side Pivot Engine	2680
Available configurations in Server-side application.....	2682
Pivot chart in EJ2 JavaScript Pivotview control.....	2700
Data binding.....	2702
Chart Types	2702
Accumulation Charts.....	2704
Field List	2716
Grouping Bar	2717
Single Axis	2720
Multiple Axis	2722

Series Customization.....	2729
Axis Customization.....	2734
Legend Customization.....	2737
User Interaction	2740
Export.....	2745
Print.....	2747
Drill down in EJ2 JavaScript Pivotview control.....	2748
Drill down and drill up.....	2748
Drill position.....	2749
Expand all	2749
Expand all headers for specific fields.....	2751
Expand all except specific member(s).....	2752
Expand specific member(s)	2754
Event	2755
Data Shaping	2762
Aggregation in EJ2 JavaScript Pivotview control.....	2762
Calculated field in EJ2 JavaScript Pivotview control	2778
Grouping in EJ2 JavaScript Pivotview control	2792
Filtering in EJ2 JavaScript Pivotview control.....	2792
Sorting in EJ2 JavaScript Pivotview control.....	2792
Drill through in EJ2 JavaScript Pivotview control.....	2792
Editing in EJ2 JavaScript Pivotview control	2792
Data Formatting	2793
Number formatting in EJ2 JavaScript Pivotview control.....	2793
Conditional formatting in EJ2 JavaScript Pivotview control	2793
Report Manipulation.....	2793
Grouping bar in EJ2 JavaScript Pivotview control.....	2793
Field list in EJ2 JavaScript Pivotview control.....	2793
Defer update in EJ2 JavaScript Pivotview control.....	2832
Performance	2836
Virtual scrolling in EJ2 JavaScript Pivotview control	2836
Paging in EJ2 JavaScript Pivotview control	2844
State persistence in EJ2 JavaScript Pivotview control	2862
Save and Load Pivot Layout	2863
Row and column in EJ2 JavaScript Pivotview control	2864

Width and Height	2864
Row Height.....	2866
Column Width	2868
Reorder	2871
Column Resizing.....	2872
Text Wrap.....	2874
Text Align	2876
AutoFit.....	2878
Grid Lines	2882
Selection.....	2884
Clip Mode	2894
Cell Template	2896
Events.....	2899
Summary customization in EJ2 JavaScript Pivotview control	2906
Show or hide grand totals	2906
Show grand totals at top or bottom	2907
Show or hide sub-totals	2909
Show or hide sub-totals for specific fields	2910
Show sub-totals at top or bottom.....	2912
Show or hide totals using toolbar	2915
Hyper link in EJ2 JavaScript Pivotview control.....	2916
Hyperlink for all cells.....	2917
Hyperlink for row headers	2918
Hyperlink for column headers.....	2920
Hyperlink for value cells.....	2921
Hyperlink for summary cells	2923
Condition based hyperlink	2924
Header based hyperlink	2926
Event	2928
See Also	2929
Tool bar in EJ2 JavaScript Pivotview control	2929
Built-in Toolbar Options.....	2930
Show desired chart types in the dropdown menu	2933
Switch the chart to multiple axes	2935
Show or hide legend	2936

Adding custom option to the toolbar	2937
Save and load report as a JSON file.....	2943
Save and load reports to a SQL database	2945
Events.....	2966
See Also	2976
Tool tip in EJ2 JavaScript Pivotview control.....	2976
Tooltip Template.....	2978
Css customization in EJ2 JavaScript Pivotview control	2980
Hiding Axis.....	2980
PivotTable .e-group-columns {.....	2980
PivotTable .e-group-filters {	2980
PivotTable_PivotFieldList_Wrapper .e-field-list-columns{	2980
PivotTable_PivotFieldList_Wrapper .e-field-list-values{.....	2980
Text Alignment.....	2982
Customize header, value and summary cell style.....	2983
Printing and Exporting	2985
Print.....	2985
Excel export in EJ2 JavaScript Pivotview control	2988
Pdf export in EJ2 JavaScript Pivotview control	3010
Globalization and localization in EJ2 JavaScript Pivotview control.....	3047
Internationalization.....	3047
Localization	3055
Right-to-left (RTL).....	3059
See Also	3061
Accessibility in Javascript Pivotview component.....	3061
WAI-ARIA attributes.....	3062
Keyboard interaction	3063
Ensuring accessibility	3070
See also	3074
Ej1 api migration in EJ2 JavaScript Pivotview control.....	3074
Data Binding.....	3074
Aggregation.....	3075
Number Format.....	3075
Summary Customization	3075
Drill operation	3076

Field List	3076
Grouping Bar	3077
Filtering	3077
Maximum node limit in member editor	3077
No Data Items	3078
Excel-like filtering.....	3078
Sorting	3078
Value Sorting.....	3078
Calculated Field.....	3079
Paging.....	3079
Conditional Formatting.....	3079
Exporting	3080
Grid Customization	3080
Drill Through	3081
Cell Editing	3081
Hyperlink	3082
Defer Layout Update.....	3083
Accessibility.....	3083
Common.....	3083
How To	3084
Switching older themes style in EJ2 JavaScript Pivotview control.....	3084
Customize number date and time values in EJ2 JavaScript Pivotview control.....	3087
Customize the icons for pivot table in EJ2 JavaScript Pivotview control.....	3091
PivotView_PivotFieldList .e-icons.e-toggle-field-list::before {.....	3091
Configure data grid options on editing mode in EJ2 JavaScript Pivotview control.....	3092
Refresh the field list in EJ2 JavaScript Pivotview control.....	3094
Hide empty headers in EJ2 JavaScript Pivotview control.....	3096
Customizing loading indicator in EJ2 JavaScript Pivotview control	3098
Changing the pivotview component minimum height in EJ2 JavaScript Pivotview control	3099
Chart based on pivot table selection in EJ2 JavaScript Pivotview control	3101
Drill through grid cell edit type in EJ2 JavaScript Pivotview control.....	3104
Show field list when pivot table empty in EJ2 JavaScript Pivotview control	3106
Apply custom style to pivot cells	3107
Show tooltip for row and column headers	3110
Hide specific columns in pivot table	3112

Export table and chart into the same document using toolbar.....	3114
Add custom aggregation type to the menu in EJ2 JavaScript Pivotview control.....	3116
Convert complex JSON to flat JSON and assign it to the pivot table in EJ2 JavaScript Pivotview control.....	3118
Load desired report from the report list as default in EJ2 JavaScript Pivotview control.....	3123
Display string value to pivot table values in EJ2 JavaScript Pivotview control	3127
Predefined Dialogs	3129
Draggable in EJ2 JavaScript Predefined dialogs control	3129
Alert drag	3129
Confirm drag	3130
Prompt drag	3131
Animation in EJ2 JavaScript Predefined dialogs control	3132
Alert animation	3132
Confirm animation	3133
Prompt animation	3134
Position in EJ2 JavaScript Predefined dialogs control.....	3135
Alert position.....	3136
Confirm position	3137
Prompt position	3138
Dimension in EJ2 JavaScript Predefined dialogs control.....	3139
Alert dimension.....	3139
Confirm dimension.....	3140
Prompt dimension	3141
Max-width and max-height.....	3142
Min-width and min-height.....	3143
Customization in EJ2 JavaScript Predefined dialogs control.....	3145
Alert action button.....	3145
Confirm action buttons	3146
Prompt action buttons.....	3147
Show or hide dialog close button	3148
Alert dialog close button.....	3148
Confirm dialog close button.....	3149
Prompt dialog close button.....	3150
Customize dialog content	3152
Progressbar	3153

Types in EJ2 JavaScript Progressbar control	3153
Linear.....	3153
Circular	3155
Tool tip in EJ2 JavaScript Progressbar control	3156
Format.....	3157
Customization	3158
States in EJ2 JavaScript Progressbar control.....	3160
Determinate	3160
Indeterminate	3161
Buffer	3162
Customization in EJ2 JavaScript Progressbar control	3163
Segments.....	3163
Thickness.....	3164
Radius.....	3165
InnerRadius	3166
Progress color and track color	3167
Annotation in EJ2 JavaScript Progressbar control	3168
Annotation	3168
Label.....	3169
Animation in EJ2 JavaScript Progressbar control.....	3170
Range in EJ2 JavaScript Progressbar control.....	3172
Theme in EJ2 JavaScript Progressbar control	3173
Events in EJ2 JavaScript Progressbar control.....	3174
Accessibility in EJ2 JavaScript Progress bar control	3176
WAI-ARIA attributes.....	3177
Keyboard interaction	3177
Ensuring accessibility	3177
See also	3177
ProgressButton	3178
Spinner and progress in EJ2 JavaScript Progress button control.....	3178
Progress.....	3179
See Also.....	3190
Accessibility in EJ2 JavaScript Progress button control	3190
WAI-ARIA attributes.....	3191
Keyboard interaction	3191

Ensuring accessibility	3192
See also	3192
How To	3192
Change the text content and styles of the progressbutton during progress in EJ2 JavaScript Progress button control.....	3192
Customize progress using cssclass in EJ2 JavaScript Progress button control.....	3195
Hide spinner in EJ2 JavaScript Progress button control.....	3196
Trace events of progress button in EJ2 JavaScript Progress button control	3199
Query Builder	3201
Columns in EJ2 JavaScript Query builder control.....	3201
Auto generation	3201
Labels	3202
Operators	3202
Step	3203
Format.....	3203
Validation	3205
Data binding in EJ2 JavaScript Query builder control	3207
Local data	3208
Remote data.....	3210
Support with Data Manager	3216
Complex Data Binding.....	3219
Filtering in EJ2 JavaScript Query builder control	3222
Model binding in EJ2 JavaScript Query builder control	3225
Templates in EJ2 JavaScript Query builder control.....	3227
Header Template	3227
Column Template.....	3231
Rule Template	3236
Import export in EJ2 JavaScript Query builder control	3240
Initial rendering.....	3241
Post rendering.....	3243
Exporting	3247
Exporting to JSON	3247
Exporting to SQL.....	3247
Global local in EJ2 JavaScript Query builder control.....	3250
Style and appearance in EJ2 JavaScript Query builder control.....	3253

Accessibility in EJ2 JavaScript Query builder control	3254
WAI-ARIA attributes.....	3255
Keyboard interaction	3255
Ensuring accessibility	3255
See also	3255
How To	3255
Display mode in EJ2 JavaScript Query builder control.....	3255
Sort columns in EJ2 JavaScript Query builder control	3257
Restrict groups in EJ2 JavaScript Query builder control	3260
State persistence in EJ2 JavaScript Query builder control.....	3262
Rtl in EJ2 JavaScript Query builder control	3264
Summary view in EJ2 JavaScript Query builder control.....	3267

Gantt

Module in EJ2 JavaScript Gantt control

The modules that are available in Gantt are as follows.

Module	Description
----- -----	
Sort	Inject this module to use sorting feature.
Filter	Inject this module to use filtering feature.
Reorder	Inject this module to use reorder feature.
ExcelExport	Inject this module to use excel export feature.
PdfExport	Inject this module to use PDF export feature.
RowDD	Inject this module to use row drag and drop feature.
Resize	Inject this module to use resize feature.
Toolbar	Inject this module to use toolbar feature.
Edit	Inject this module is use editing feature.
Selection	Inject this module to use selection feature.
DayMarkers	Inject this module to use event markers.
ContextMenu	Inject this module to use context menu feature.
ColumnMenu	Inject this module to use column menu feature.
VirtualScroll	Inject this module to use virtual scroll feature.
CriticalPath	Inject this module to use critical path feature.

These modules should be injected into the Gantt using the **Gantt.Inject** method.

Data binding in EJ2 JavaScript Gantt control

The Gantt control uses **DataManager** for binding the data source, which supports both RESTful JSON data services and local JavaScript object array. The [dataSource](#) property can be assigned either with the instance of DataManager or JavaScript object array collection. The Gantt control supports binding two types of data:

- Local data
- Remote data

Local data

To bind local data to Gantt, you can assign a JavaScript object array to the [dataSource](#) property. The local data source can also be provided as an instance of the **DataManager**.

In local data binding, the data source for rendering the Gantt control is retrieved from the same application locally.

The following are the two types of data binding possible with the Gantt control:

- Hierarchical data binding.
- Self-referential data binding (Flat data).

Hierarchical data binding

The [child](#) property is used to map the child records in hierarchical data.

The following code example shows how to bind the hierarchical local data into the Gantt control.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
let HierarchyData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 }
        ]
    },
];
let gantt: Gantt = new Gantt({
    dataSource: HierarchyData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Self-referential data binding (Flat data)

The Gantt control can be bound with self-referential data by mapping the data source field values to the [id](#) and [parentID](#) properties.

- ID field: This field contains unique values used to identify each individual task and it is mapped to the [id](#) property.
- Parent ID field: This field contains values that indicate parent tasks and it is mapped to the [parentID](#) property.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
let SelfReferenceData: Object[] = [
  { TaskID: 1, TaskName: 'Project Initiation', StartDate: new
Date('04/02/2019'), EndDate: new Date('04/21/2019') },
  { TaskID: 2, TaskName: 'Identify Site location', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, ParentId: 1 },
  { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, ParentId: 1 },
  { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, ParentId: 1 },
  { TaskID: 5, TaskName: 'Project Estimation', StartDate: new
Date('04/02/2019'), EndDate: new Date('04/21/2019') },

```



```

        { TaskID: 6, TaskName: 'Develop floor plan for estimation',
        StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50, ParentId:5 },
        { TaskID: 7, TaskName: 'List materials', StartDate: new
        Date('04/04/2019'), Duration: 3, Progress: 50, ParentId:5 },
        { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
        Date('04/04/2019'), Duration: 3, Progress: 50, ParentId:5 }
    ];
    let gantt: Gantt = new Gantt({
        dataSource: SelfReferenceData,
        height: '450px',
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            parentID: 'ParentId'
        }
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
    rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Remote data

To bind remote data to the Gantt component, assign service data as an instance of **DataManager** to the [dataSource](#) property.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
let dataSource: DataManager = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
    services/api/GanttData',
    adaptor: new WebApiAdaptor,
    crossDomain: true
});
let gantt: Gantt = new Gantt({
    dataSource: dataSource,
    height: '450px',
    treeColumnIndex: 1,
    taskFields: {
        id: 'TaskId',
        name: 'TaskName',
        startDate: 'StartDate',
        progress: 'Progress',
        duration: 'Duration',
        dependency: 'Predecessor',
        child: 'SubTasks'
    }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
    rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

URL Adaptor

In Gantt, we can fetch data from SQL database using ADO.NET Entity Data Model and update the changes on CRUD action to the server by using DataManager support. To communicate with the remote data we are using UrlAdaptor of DataManager property to call the server method and get back resultant data in JSON format. We can know more about UrlAdaptor from [here](#).

Please refer the [link](#) to create the ADO.NET Entity Data Model in Visual studio,

We can define data source for Gantt as instance of DataManager using url property of DataManager. Please Check the below code snippet to assign data source to Gantt.

```

`ts
import { Gantt } from '@syncfusion/ej2-gantt';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
let dataSource: DataManager = new DataManager({
url: '/Home/UrlDatasource',
adaptor: new UrlAdaptor
});
let gantt: Gantt = new Gantt({
dataSource: dataSource,
height: '450px',
treeColumnIndex: 1,
taskFields: {
id: 'TaskId',
name: 'TaskName',
startDate: 'StartDate',
progress: 'Progress',
duration: 'Duration',
dependency: 'Predecessor',
child: 'SubTasks'
}
});
gantt.appendTo('#Gantt');

```

```

`ts
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult UrlDatasource(DataManagerRequest dm)
{
    List<GanttData>DataList = db.GanttDatas.ToList();
    var count = DataList.Count();
    return Json(new { result = DataList, count = count });
}
`

```

Load child on demand

To render child records on demand, assign a remote service URL in the instance of DataManager to the Url property. To interact with the remote data source, provide the endpoint URL and also define the [hasChildMapping](#) property in taskFields of Gantt Chart.

The `hasChildMapping` property maps the field name in the data source, which denotes whether the current record holds any child records. This is useful internally to show expand icon while binding child data on demand.

When [loadChildOnDemand](#) is disabled, all the root nodes are rendered in a collapsed state at initial load. On expanding the root node, the child nodes will be loaded from the remote server.

When `enableVirtualization` is enabled and `loadChildOnDemand` is disabled, only the current viewport root nodes are rendered in a collapsed state.

When a root node is expanded, its child nodes are rendered and maintained in a collection locally, such that on consecutive expand/collapse actions on the root node, the child nodes are loaded locally instead of from the remote server.

When the `loadChildOnDemand` is enabled, parent records are rendered in an expanded state.

```

`ts
import { Gantt, VirtualScroll, Selection } from '@syncfusion/ej2-gantt';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
import './App.css';
Gantt.Inject(Selection, VirtualScroll);
let dataSource: DataManager = new DataManager({
    url: 'https://ej2services.syncfusion.com/js/development/api/GanttLoadOnDemand',
    adaptor: new WebApiAdaptor,
    crossDomain: true
});

```

```
let gantt: Gantt = new Gantt({
  dataSource: dataSource,
  loadChildOnDemand: false,
  taskFields: {
    id: 'taskId',
    name: 'taskName',
    startDate: 'startDate',
    endDate: 'endDate',
    duration: 'duration',
    progress: 'progress',
    hasChildMapping: 'isParent',
    parentID: 'parentID'
  },
  columns: [
    { field: 'taskId', headerText: 'Task ID' },
    { field: 'taskName', headerText: 'Task Name', allowReordering: false },
    { field: 'startDate', headerText: 'Start Date', allowSorting: false },
    { field: 'duration', headerText: 'Duration', allowEditing: false },
    { field: 'progress', headerText: 'Progress', allowFiltering: false },
  ],
  allowSelection: true,
  enableVirtualization: true,
  splitterSettings: {
    columnIndex: 3,
  },
  tooltipSettings: {
    showTooltip: true
  },
  highlightWeekends: true,
  timelineSettings: {
    showTooltip: true,
    topTier: {
      unit: 'Week',
```

```

format: 'dd/MM/yyyy'
},
bottomTier: {
unit: 'Day',
count: 1
}
},
treeColumnIndex: 1,
taskbarHeight: 20,
rowHeight: 40,
height: '460px',
projectStartDate: new Date('01/02/2000'),
projectEndDate: new Date('12/01/2002'),
});
gant.appendTo('#Gantt');
`

```

The following code example describes handling of Load on demand at server end.

```

`ts
public object Get()
{
DataOperations operation = new DataOperations();
var queryString = Request.Query;
if (tree.Count == 0)
tree = TreeData.GetTree();
if (queryString.Keys.Contains("$filter") && !queryString.Keys.Contains("$stop"))
{
StringValues filter;
queryString.TryGetValue("$filter", out filter);
int? fltr;
if (filter[0].ToString().Split("eq")[1] == " null")
{
fltr = null;
}
}
}

```

```
else
{
    fltr = Int32.Parse(filter[0].ToString().Split("eq")[1]);
}

IQueryable<TreeData> data1 = tree.Where(f => f.parentID == fltr).AsQueryable();
return new { result = data1.ToList(), count = data1.Count() };
}

StringValues expand;
queryString.TryGetValue("$expand", out expand);

if (queryString.Keys.Contains("$expand")) // setting the ExpandStateMapping property whether is true
or false
{
    if (expand[0].ToString().Split(",")[0] == "ExpandingAction")
    {
        var val = TreeData.GetTree().Where(ds => ds.taskId ==
int.Parse(expand[0].ToString().Split(",")[1])).FirstOrDefault();
        val.IsExpanded = true;
    }
    else if (expand[0].ToString().Split(",")[0] == "CollapsingAction")
    {
        var val = TreeData.GetTree().Where(ds => ds.taskId ==
int.Parse(expand[0].ToString().Split(",")[1])).FirstOrDefault();
        val.IsExpanded = false;
    }
}

List<TreeData> data = tree.ToList();
if (queryString.Keys.Contains("$select"))
{
    data = (from ord in tree
select new TreeData
{
    parentID = ord.parentID
}
).ToList();
}
```

```

return data;
}
data = data.Where(p => p.parentID == null).ToList();
int count = data.Count;
if (queryString.Keys.Contains("$inlinecount"))
{
    StringValues Skip;
    StringValues Take;
    StringValues loadchild;
    int skip = (queryString.TryGetValue("$skip", out Skip)) ? Convert.ToInt32(Skip[0]) : 0;
    int top = (queryString.TryGetValue("$top", out Take)) ? Convert.ToInt32(Take[0]) : data.Count();
    var GroupData = TreeData.GetTree().ToList().GroupBy(rec => rec.parentID)
    .Where(g => g.Key != null).ToDictionary(g => g.Key?.ToString(), g => g.ToList());
    foreach (var Record in data.ToList())
    {
        if (GroupData.ContainsKey(Record.taskId.ToString()))
        {
            var ChildGroup = GroupData[Record.taskId.ToString()];
            if (ChildGroup?.Count > 0)
            AppendChildren(ChildGroup, Record, GroupData, data);
        }
    }
    if (expand.Count > 0 && expand[0].ToString().Split(",")[0] == "CollapsingAction")
    {
        string IdMapping = "taskId";
        List<WhereFilter> CollapseFilter = new List<WhereFilter>();
        CollapseFilter.Add(new WhereFilter() { Field = IdMapping, value = expand[0].ToString().Split(",")[1],
        Operator = "equal" });
        var CollapsedParentRecord = operation.PerformFiltering(data, CollapseFilter, "and");
        var index = data.Cast<object>().ToList().IndexOf(CollapsedParentRecord.Cast<object>().ToList()[0]);
        skip = index;
    }
    else if (expand.Count > 0 && expand[0].ToString().Split(",")[0] == "ExpandingAction")

```



```

{
string IdMapping = "taskId";
List<WhereFilter> ExpandFilter = new List<WhereFilter>();
ExpandFilter.Add(new WhereFilter() { Field = IdMapping, value = expand[0].ToString().Split(",")[1],
Operator = "equal" });
var ExpandedParentRecord = operation.PerformFiltering(data, ExpandFilter, "and");
var index = data.Cast<object>().ToList().IndexOf(ExpandedParentRecord.Cast<object>().ToList()[0]);
skip = index;
}
return new { result = data.Skip(skip).Take(top), count = data.Count };
}
else
{
return TreeData.GetTree();
}
}

private void AppendChildren(List<TreeData> ChildRecords, TreeData ParentItem, Dictionary<string,
List<TreeData>> GroupData, List<TreeData> data)
{
var queryString = Request.Query;
string TaskId = ParentItem.taskId.ToString();
var index = data.IndexOf(ParentItem);
foreach (var Child in ChildRecords)
{
string ParentId = Child.parentID.ToString();
if (TaskId == ParentId && (bool)ParentItem.IsExpanded)
{
if (data.IndexOf(Child) == -1)
((IList)data).Insert(++index, Child);
if (GroupData.ContainsKey(Child.taskId.ToString()))
{
var DeepChildRecords = GroupData[Child.taskId.ToString()];
if (DeepChildRecords?.Count > 0)
AppendChildren(DeepChildRecords, Child, GroupData, data);
}
}
}
}

```

```
}  
}  
}  
}  
  
// GET: api/Orders/  
[HttpGet("{id}", Name = "Get")]  
public string Get(int id)  
{  
    return "value";  
}  
  
[HttpPost]  
public object Post([FromBody] TreeData[] value)  
{  
    //handle insert action  
    for (var i = 0; i < value.Length; i++)  
    {  
        tree.Insert(0, value[i]);  
    }  
    return value;  
}  
  
///// PUT: api/Orders  
[HttpPut]  
public object Put([FromBody] TreeData[] value)  
{  
    //handle edit action  
    if (value.Length == 1 && value[0].isParent == true)  
    {  
        UpdateDependentRecords(value[0]);  
    }  
    for (var i = 0; i < value.Length; i++) {  
        var ord = value[i];  
        TreeData val = tree.Where(or => or.taskId == ord.taskId).FirstOrDefault();  
        val.taskId = ord.taskId;
```

```
val.taskName = ord.taskName;
val.endDate = ord.endDate;
val.startDate = ord.startDate;
val.duration = ord.duration;
val.predecessor = ord.predecessor;
}
return value;
}

private void UpdateDependentRecords(TreeData ParentItem)
{
    var data = tree.Where(p => p.parentID == ParentItem.taskId).ToList();
    var previousData = tree.Where(p => p.taskId == ParentItem.taskId).ToList();
    var previousStartDate = previousData[0].startDate;
    var previousEndDate = previousData[0].endDate;
    double sdiff = (double)GetTimeDifference((DateTime)previousStartDate,
        (DateTime)ParentItem.startDate);
    double ediff = (double)GetTimeDifference((DateTime)previousEndDate,
        (DateTime)ParentItem.endDate);
    GetRootChildRecords(ParentItem);
    for(var i=0; i<ChildRecords.Count;i++)
    {
        ChildRecords[i].startDate = ((DateTime)ChildRecords[i].startDate).AddSeconds(sdiff);
        ChildRecords[i].endDate = ((DateTime)ChildRecords[i].endDate).AddSeconds(ediff);
    }
}

private void GetRootChildRecords(TreeData ParentItem)
{
    var currentchildRecords = tree.Where(p => p.parentID == ParentItem.taskId).ToList();
    for (var i = 0; i < currentchildRecords.Count; i++) {
        var currentRecord = currentchildRecords[i];
        ChildRecords.Add(currentRecord);
        if (currentRecord.isParent == true)
        {
            GetRootChildRecords(currentRecord);
        }
    }
}
```

```

}
}
}
public object GetTimeDifference(DateTime sdate, DateTime edate)
{
    return new DateTime(edate.Year, edate.Month, edate.Day, edate.Hour, edate.Minute, edate.Second,
        DateTimeKind.Utc).Subtract(new DateTime(sdate.Year, sdate.Month, sdate.Day, sdate.Hour,
            sdate.Minute, sdate.Second, DateTimeKind.Utc)).TotalSeconds;
}
// DELETE: api/ApiWithActions
[HttpDelete("{id:int}")]
[Route("Orders/{id:int}")]
public object Delete(int id)
{
    //handle delete action
    tree.Remove(tree.Where(or => or.taskId == id).FirstOrDefault());
    return Json(id);
}
public class CRUDModel<T> where T : class
{
    public TreeData Value;
    public int Key { get; set; }
    public int RelationalKey { get; set; }
    public List<TreeData> added { get; set; }
    public List<TreeData> changed { get; set; }
    public List<TreeData> deleted { get; set; }
}
public class TreeData
{
    public static List<TreeData> tree = new List<TreeData>();
    [System.ComponentModel.DataAnnotations.Key]
    public int taskId { get; set; }
    public string taskName { get; set; }
    public DateTime startDate { get; set; }
}

```

```
public DateTime endDate { get; set; }
public string duration { get; set; }
public int progress { get; set; }
public int? parentID { get; set; }
public string predecessor { get; set; }
public bool? isParent { get; set; }
public bool? IsExpanded { get; set; }
public static List<TreeData> GetTree()
{
    if (tree.Count == 0)
    {
        Random rand = new Random();
        var x = 0;
        int duration = 0;
        DateTime startDate = new DateTime(2000, 1, 3, 08, 00, 00);
        for (var i = 1; i <= 50; i++)
        {
            startDate = startDate.AddDays(i == 1 ? 0 : 7);
            DateTime childStartDate = startDate;
            TreeData Parent = new TreeData()
            {
                taskId = ++x,
                taskName = "Task " + x,
                startDate = startDate,
                endDate = childStartDate.AddDays(26),
                duration = "20",
                progress = rand.Next(100),
                predecessor = null,
                isParent = true,
                parentID = null,
                IsExpanded = false
            };
            tree.Add(Parent);
        }
    }
}
```

```

for (var j = 1; j <= 4; j++)
{
    childStartDate = childStartDate.AddDays(j == 1 ? 0 : duration + 2);
    duration = 5;
    tree.Add(new TreeData()
    {
        taskId = ++x,
        taskName = "Task " + x,
        startDate = childStartDate,
        endDate = childStartDate.AddDays(5),
        duration = duration.ToString(),
        progress = rand.Next(100),
        parentID = Parent.taskId,
        predecessor = (j > 1 ? (x - 1) + "FS" : ""),
        isParent = false,
        IsExpanded = false
    });
}
}
}
return tree;
}
}
,

```

Limitations

- Filtering, sorting and searching are not supported in load on demand.
- Only Self-Referential type data is supported with remote data binding in Gantt Chart.
- Load-on-demand supports only the validated data source

Sending additional parameters to the server

We can pass additional parameters using [addParams](#) method of [Query](#) class.

In server side we have inherited and shown the additional parameter value in Syncfusion DataManager class itself. We pass an additional parameter in load time using [load](#) event. We can also pass additional parameter to the CRUD model. Please Check the below code snippet to send additional parameter to Gantt.

```
`ts
import { Gantt, Edit, Toolbar } from '@syncfusion/ej2-gantt';
import { DataManager, UrlAdaptor, Query } from '@syncfusion/ej2-data';
Gantt.Inject(Edit, Toolbar);
let dataSource: DataManager = new DataManager({
  url: 'http://localhost:50039/Home/UrlDatasource',
  adaptor: new UrlAdaptor,
  batchUrl: 'http://localhost:50039/Home/BatchSave',
});
let gantt: Gantt = new Gantt({
  dataSource: dataSource,
  height: '450px',
  treeColumnIndex: 1,
  taskFields: {
    id: 'taskID',
    name: 'taskName',
    startDate: 'startDate',
    endDate: 'endDate',
    duration: 'duration',
    progress: 'progress',
    parentID: 'parentID',
  },
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
  },
  toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll', 'CollapseAll'],
  load: function (args) {
    this.query = new Query().addParams('ej2Gantt', "test");
  }
});
gantt.appendTo('#Gantt');
```

```
`ts
namespace URLAdaptor.Controllers
{
    public class HomeController : Controller
    {
        ...///
        //inherit the class to show age as property of DataManager
        public class Test : DataManagerRequest
        {
            public string ej2Gantt { get; set; }
        }
        public ActionResult UrlDatasource([FromBody]Test dm)
        {
            if (DataList == null)
            {
                ProjectData datasource = new ProjectData();
                DataList = datasource.GetUrlDataSource();
            }
            var count = DataList.Count();
            return Json(new { result = DataList, count = count }, JsonRequestBehavior.AllowGet);
        }
        ...///
        public class ICRUDModel<T> where T : class
        {
            public object key { get; set; }
            public T value { get; set; }
            public List<T> added { get; set; }
            public List<T> changed { get; set; }
            public List<T> deleted { get; set; }
            public IDictionary<string, object> @params { get; set; }
        }
        ...///
```



```

}
}
,

```

You can find the full sample from [here](#).

Handling HTTP error

During server interaction from the Gantt, some server-side exceptions may occur, and you can acquire those error messages or exception details in client-side using the [actionFailure](#) event.

The argument passed to the `actionFailure` event contains the error details returned from the server.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { DataManager } from '@syncfusion/ej2-data';
let dataSource: DataManager = new DataManager({
    url: 'http://some.com/invalidUrl',
});
let gantt: Gantt = new Gantt({
    dataSource: dataSource,
    height: '450px',
    treeColumnIndex: 1,
    taskFields: {
        id: 'taskID',
        name: 'taskName',
        startDate: 'startDate',
        endDate: 'endDate',
        duration: 'duration',
        progress: 'progress',
        parentID: 'parentID',
    },
    actionFailure: (e) => {
        let span: HTMLElement = document.createElement('span');
        gantt.element.parentNode.insertBefore(span, gantt.element);
        span.style.color = '#FF0000';
        span.innerHTML = 'Server exception: 404 Not found';
    },
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Binding with Ajax

You can use Gantt [dataSource](#) property to bind the data source to Gantt from external Ajax request. In the below code we have fetched the data source from the server with the help of Ajax request and provided that to `dataSource` property by using [onSuccess](#) event of the Ajax.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { Ajax } from '@syncfusion/ej2-base';
let gantt: Gantt = new Gantt({
    height: '450px',
    treeColumnIndex: 1,
    taskFields: {
        id: 'TaskId',
        name: 'TaskName',
        startDate: 'StartDate',
        progress: 'Progress',
        duration: 'Duration',
        dependency: 'Predecessor',
        child: 'SubTasks'
    },
    projectStartDate: new Date('02/24/2019'),
    projectEndDate: new Date('07/20/2019')
});
gantt.appendTo('#Gantt');
let button: HTMLElement = document.createElement('button');
button.textContent = 'Bind Data';
gantt.element.parentNode.insertBefore(button, gantt.element);
button.addEventListener("click", function(e) {
    let ajax = new Ajax("https://ej2services.syncfusion.com/production/web-services/api/GanttData", "GET");
    gantt.showSpinner();
    ajax.send();
    ajax.onSuccess = function (data: string) {
        gantt.hideSpinner();
        gantt.dataSource = (JSON.parse(data)).Items;
        gantt.refresh();
    };
});

```

```
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server side crud actions.

Split task

The **Split-task** feature allows you to split a task or interrupt the work during planned or unforeseen circumstances. We can split the task either in load time or dynamically, by defining the segments either in hierarchical or self-referential way.

Hierarchical

To split a task at load time in hierarchical way, we need to define the segment details in datasource and this field should be mapped by using the [taskFields.segments](#) property.

```
`ts
```

```
[
{
```

```
TaskID: 1, TaskName: 'Identify Site location', StartDate: new Date('04/02/2019'), Duration: 4, Progress:
50,
```

Segments: [

{ StartDate: new Date("04/02/2019"), Duration: 2 },

{ StartDate: new Date("04/04/2019"), Duration: 2 }

]

}

]

,

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: "450px",
  taskFields: {
    id: "TaskID",
    name: "TaskName",
    startDate: "StartDate",
    endDate: "EndDate",
    duration: "Duration",
    progress: "Progress",
    child: "subtasks",
    segments: "Segments"
  },
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
```

```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Self-referential

We can also define segment details as a flat data and this collection can be mapped by using [segmentData](#) property. The segment id field of this collection is mapped by using the [taskFields.segmentId](#) property.

`ts

```

taskFields: {
    segmentId: "segmentId"
},
segmentData: [
    { segmentId: 1, StartDate: new Date("02/04/2019"), Duration: 2 },
    { segmentId: 1, StartDate: new Date("02/05/2019"), Duration: 5 },
    { segmentId: 4, StartDate: new Date("04/02/2019"), Duration: 2 },
    { segmentId: 4, StartDate: new Date("04/04/2019"), Duration: 2 }
],
`

```

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { SplitTaskData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: SplitTaskData,
    height: "450px",
    taskFields: {
        id: "TaskID",
        name: "TaskName",
        startDate: "StartDate",
        endDate: "EndDate",
        duration: "Duration",
        progress: "Progress",
        child: "subtasks",
        segmentId: "segmentId"
    },
    segmentData: [
        { segmentId: 2, StartDate: new Date("04/02/2019"), Duration: 2 },
        { segmentId: 2, StartDate: new Date("04/04/2019"), Duration: 2 },
        { segmentId: 4, StartDate: new Date("04/02/2019"), Duration: 2 },
        { segmentId: 4, StartDate: new Date("04/04/2019"), Duration: 2 }
    ],
}

```

```
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: Segment id field contains id of a task which should be splitted at load time.

Limitations

Gantt has the support for both Hierarchical and Self-Referential data binding. When rendering the Gantt control with SQL database, we suggest you to use the Self-Referential data binding to maintain the parent-child relation. Because the complex json structure is very difficult to manage it in SQL tables, we need to write a complex queries and we have to write a complex algorithm to find out the proper record details while updating/deleting the inner level task in Gantt data source. We cannot implement both data binding for Gantt control and this is not a recommended way. If both child and parentID are mapped, the records will not render properly because, when task id of a record defined in the hierarchy structure is assigned to parent id of another record, in such case the records will not properly render. As the self-referential will search the record with particular id in flat data only, not in the inner level of records. If we map the parentID field, it will be prioritized and Gantt will be rendered based on the parentID values.

Immutable in EJ2 JavaScript Gantt control

The immutable mode optimizes the Gantt re-rendering performance by using the object reference and [deep compare](#) concept. When performing the Gantt actions, it will only re-render the modified or newly added rows and prevent the re-rendering of the unchanged rows.

To enable this feature, you have to set the [enableImmutableMode](#) property as **true**.

This feature uses the primary key value for data comparison. So, you need to provide the [isPrimaryKey](#) column.

INDEX.TS

```
import { Gantt, Toolbar, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    enableImmutableMode: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar: ['Add', 'Edit', 'Delete', 'Update',
'Cancel', 'Indent', 'Outdent'],
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt .e-gantt-chart .e-custom-holiday {
      background-color:#e82869;
```

```

    }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations

The following features are not supported in the immutable mode:

- Column reorder
- Virtualization

Virtual scroll in EJ2 JavaScript Gantt control

Virtual Scroll support in Gantt allows you to load large amount of data without performance degradation. To enable Virtual Scrolling, you need to inject `VirtualScroll` module in Gantt.

Row virtualization

Row virtualization allows you to load and render a large number of tasks in Gantt with effective performance. In this mode, all tasks are fetched initially from the datasource and rendered in the DOM within a compact viewport area.

The number of records displayed in the Gantt is determined by the height.

This mode can be enable by setting the `enableVirtualization` property to `true`.

INDEX.TS

```

import { Gantt, Selection, VirtualScroll } from '@syncfusion/ej2-gantt';
import { virtualData } from 'datasource.ts';
Gantt.Inject(Selection, VirtualScroll);
let gantt: Gantt = new Gantt({
    dataSource: virtualData,
    treeColumnIndex: 1,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',

```



```

        duration: 'Duration',
        progress: 'Progress',
        parentID: 'parentID'
    },
    enableVirtualization: true,
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName' },
        { field: 'StartDate' },
        { field: 'Duration' },
        { field: 'Progress' },
    ],
    allowSelection: true,
    gridLines: 'Both',
    height: '450px',
    splitterSettings: {
        columnIndex: 2
    },
    labelSettings: {
        taskLabel: 'Progress'
    }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

Timeline virtualization

Timeline virtualization allows you to load a data source having large timespan with high performance. Initially, it renders the timeline with thrice the width of the gantt element, while other timeline cells render on-demand during horizontal scrolling.

This mode can be enable by setting the [enableTimelineVirtualization](#) property to `true`.

INDEX.TS

```
import { Gantt, Selection, VirtualScroll } from '@syncfusion/ej2-gantt';
import { virtualData } from 'datasource.ts';
Gantt.Inject(Selection, VirtualScroll);
let gantt: Gantt = new Gantt({
  dataSource: virtualData,
  treeColumnIndex: 1,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    parentID: 'parentID'
  },
  enableVirtualization: true,
  columns: [
    { field: 'TaskID' },
    { field: 'TaskName' },
    { field: 'StartDate' },
    { field: 'Duration' },
    { field: 'Progress' },
  ],
  allowSelection: true,
  enableTimelineVirtualization: true,
  gridLines: 'Both',
  height: '450px',
  splitterSettings: {
    columnIndex: 2
  },
  labelSettings: {
    taskLabel: 'Progress'
  },
  projectStartDate: new Date('04/01/2019'),
  projectEndDate: new Date('12/31/2025')
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Get filtered data when virtual scrolling is enabled

While enabling virtual scroll you can get the filtered or sorted record count using `filteredResult` from the `filterModule` of the `treegrid` inside the `actionComplete` event.

INDEX.TS

```

import { Gantt, Selection, VirtualScroll, Sort, Filter } from
'@syncfusion/ej2-gantt';
import { virtualData } from 'datasource.ts';
Gantt.Inject(Selection, VirtualScroll, Sort, Filter);
let filteredCount: any = 0;
let datasetCount: any;
let gantt: Gantt = new Gantt({
    dataSource: virtualData,
    treeColumnIndex: 1,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        parentID: 'parentID'
    },
    allowSorting: true,
    allowFiltering: true,
    actionComplete: function (args: any) {
        if (args.requestType == 'filtering') {
            if (args.rows != null) {
                filteredCount =

```

```

        gantt.treeGrid.filterModule.filteredResult.length;
        let combinedMessage = `Dataset Count: ${datasetCount}
Filtered Count: ${filteredCount}`;
        const countElement = document.getElementById('count-
element');
        if (countElement) {
            countElement.textContent = combinedMessage;
        }
    },
    created: function() {
        datasetCount = gantt.flatData.length;
        let combinedMessage = `Dataset Count: ${datasetCount} Filtered
Count: ${filteredCount}`;
        const countElement = document.getElementById('count-element');
        if (countElement) {
            countElement.textContent = combinedMessage;
        }
    },
    enableVirtualization: true,
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName' },
        { field: 'StartDate' },
        { field: 'Duration' },
        { field: 'Progress' },
    ],
    allowSelection: true,
    gridLines: 'Both',
    height: '450px',
    splitterSettings: {
        columnIndex: 2
    },
    labelSettings: {
        taskLabel: 'Progress'
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
        <h1>Counts Display</h1>
        <p id="count-element"></p>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations for virtual vcroll

- Due to the element height limitation in browsers, the maximum number of records loaded is limited by the browser capacity.
- Cell selection will not be persisted.
- The number of records rendered will be determined by the `height` property.
- It is necessary to mention the height of the Gantt in pixels when enabling Virtual Scrolling.
- Virtual Scroll does not support Multi Taskbar support in Resource View.

Selection

Selection in EJ2 JavaScript Gantt control

Selection provides an option to highlight a row or a cell. It can be done using arrow keys or by scrolling down the mouse. To disable selection in the Gantt control, set the [allowSelection](#) to false.

To select data, inject the [Selection](#) module into the Gantt control.

The Gantt control supports two types of selection that can be set by using the [selectionSettings.type](#) property. They are:

- **Single**: Sets a single value by default and allows only selection of a single row or a cell.
- **Multiple**: Allows you to select multiple rows or cells. To perform the multi-selection, press and hold the CTRL key and click the desired rows or cells.

Selection mode

The Gantt control supports three types of selection modes that can be set by using the [selectionSettings.mode](#). They are:

- **Row**: Allows you to select only rows, and the row value is set by default.
- **Cell**: Allows you to select only cells.
- **Both**: Allows you to select rows and cells at the same time.

INDEX.TS

```
import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  selectionSettings: {
    mode: 'Both'
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Toggle selection

The toggle selection allows you to select and deselect a specific row or cell. To enable toggle selection, set the `enableToggle` property of the `selectionSettings` to `true`. If you click the selected row or cell, then it will be deselected and vice versa. By default, the `enableToggle` property is set to `false`.

INDEX.TS

```
import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Row',
        type: 'Multiple',
        enableToggle: true
    }
});
gantt.appendTo('#Gantt');
let toggleBtn: Button = new Button();
toggleBtn.appendTo('#toggle');
document.getElementById('toggle').addEventListener('click', () => {
    gantt.selectionSettings.enableToggle = false;
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="toggle">Disable toggle</button>
```

```

    <div id="container">
      <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clear selection

You can clear the selected cells and selected rows by using a method called [clearSelection](#). The following code example demonstrates how to clear the selected rows in Gantt Chart.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  selectionSettings: {
    mode: 'Row',
    type: 'Multiple'
  }
});
gantt.appendTo('#Gantt');
let selBtn: Button = new Button();
selBtn.appendTo('#selectRows');
let clrBtn: Button = new Button();
clrBtn.appendTo('#clearSelection');
document.getElementById('selectRows').addEventListener('click', () => {
  gantt.selectionModule.selectRows([1, 3, 5]); // passing the record index
  as array collection
});
document.getElementById('clearSelection').addEventListener('click', () => {
  gantt.clearSelection(); // Clear the selected rows
});

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```



```

<title>EJ2 Gantt</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="selectRows">Select Rows</button>
    <button id="clearSelection">Clear Selection</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Get selected row indexes and records

You can get the selected row indexes by using the [getSelectedRowIndex](#) method. And by using [getSelectedRecords](#) method, you can get the selected record details.

INDEX.TS

```

import { Gantt, Selection, RowSelectEventArgs } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Row',
        type: 'Multiple'
    }
});

```

```

    },
    rowSelected: rowSelected
  });
gantt.appendTo('#Gantt');
function rowSelected(args: RowSelectEventArgs) {
  let selectedrowindex: number[] =
gantt.selectionModule.getSelectedRowIndex(); // get the selected row
indexes.
  alert(selectedrowindex); // to alert the selected row indexes.
  let selectedrecords: Object[] =
gantt.selectionModule.getSelectedRecords(); // get the selected records.
  console.log(selectedrecords); // to print the selected records in
console window.
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple Selection based on condition

You can select multiple rows based on condition by using the [selectRows](#) method.

In the following code, the rows which contains `TaskId` value as 3 and 4 are selected at initial rendering.

INDEX.TS

```
import { Gantt, Selection, RowSelectEventArgs } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  selectionSettings: {
    mode: 'Row',
    type: 'Multiple'
  },
  dataBound: () => {
    var rowIndexes = [];
    gantt.treeGrid.grid.dataSource.forEach((data, index) => {
      if (data.TaskID === 3 || data.TaskID === 4) {
        rowIndexes.push(index);
      }
    });
    gantt.selectRows(rowIndexes);
  },
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
</body>
</html>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Touch interaction](#)

Row selection in EJ2 JavaScript Gantt control

The row selection in the Gantt control can be enabled or disabled using the [allowSelection](#) property. You can get the selected row object using the [getSelectedRecords](#) method. The following code example shows how to disable the row selection in Gantt.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowSelection: false
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Row selection is the default type of Gantt selection mode.

Selecting a row on initial load

You can select a row at the time of loading by setting the index of the row to the [selectedRowIndex](#) property. Find the following code example for details.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectedRowIndex: 3,
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selecting a row dynamically

You can also select a row dynamically using the [selectRow](#) method. The following code demonstrates how to select a row dynamically by clicking the custom button.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    }
});
gantt.appendTo('#Gantt');
let selectBtn: Button = new Button();
selectBtn.appendTo('#selectRow');
document.getElementById('selectRow').addEventListener('click', () => {
    gantt.selectionModule.selectRow(2); // passing the record index to
    select the row
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="selectRow">Select Row</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple row selection

You can select multiple rows by setting the [selectionSettings.type](#) property to **multiple**. You can select more than one row by holding down the CTRL key while selecting multiple rows. The following code example explains how to enable multiple selection in Gantt.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Row',
        type: 'Multiple'
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Selecting multiple rows dynamically

You can also select rows dynamically using the [selectRows](#) method. The following code demonstrates how to select rows dynamically by clicking the custom button.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  selectionSettings: {

```



```

        mode: 'Row',
        type: 'Multiple'
    }
});
gantt.appendTo('#Gantt');
let selBtn: Button = new Button();
selBtn.appendTo('#selectRows');
document.getElementById('selectRows').addEventListener('click', () => {
    gantt.selectionModule.selectRows([1, 2, 3]); // passing the record index
    as array collection
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <button id="selectRows">Select Rows</button>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize row selection action

While selecting a row in Gantt, the [rowSelecting](#) and [rowSelected](#) events will be triggered. The [rowSelecting](#) event will be triggered on initialization of row selection, and you can get the previously selected row and current selecting row's information, which is used to prevent selection of a particular row. The [rowSelected](#) event will be triggered on completion of row selection action, and you can get the current selected row's information through this event. The following code example demonstrates how to prevent the selection of a row using the [rowSelecting](#) event.

INDEX.TS

```
import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  rowSelecting: function (args: any) {
    if (args.data.TaskID == 4) {
      args.cancel = true;
    }
  },
  selectionSettings: {
    mode: 'Row'
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In the Gantt control, when you click an already selected row, selection will be cleared. While deselecting a row in Gantt, the [rowDeselecting](#) and [rowDeselected](#) events will occur. The [rowDeselecting](#) event will occur on initialization of deselecting row, and you can get the current deselecting row's information to prevent the deselection of particular row. The [rowDeselected](#) event will occur on completion of row deselection action, and you can get the current deselected row's information.

Cell selection in EJ2 JavaScript Gantt control

You can select a cell in the Gantt control by setting the [selectionSettings.mode](#) property to cell. You can get the selected cell information using the [getSelectedRowCellIndexes](#) method. This method returns the result as an object collection, which has `cellIndexes` and `rowIndex` information of the selected cells.

Find the code example below to enable the cell selection in Gantt.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  selectionSettings: {
    mode: 'Cell'
  }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selecting multiple cells

You can select multiple cells by setting the [selectionSettings.type](#) property to multiple and the [selectionSettings.mode](#) property to cell. Multiple cells can be selected by holding the CTRL key and selecting the cells. The following code example demonstrates how to select multiple cells.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Cell',
        type: 'Multiple'
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selecting a cell dynamically

You can select a cell dynamically using the [selectCell](#) method. Refer to the following code example for details.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    selectionSettings: {
        mode: 'Cell'
    }
});
gantt.appendTo('#Gantt');
let cellBtn: Button = new Button();
cellBtn.appendTo('#selectCell');
document.getElementById('selectCell').addEventListener('click', () => {
    gantt.selectionModule.selectCell({ cellIndex: 1, rowIndex: 1 });
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <button id="selectCell">Select Cell</button>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize cell selection action

While selecting a cell in Gantt, the [cellSelecting](#) and [cellSelected](#) event will be triggered. The [cellSelecting](#) event will be triggered on initialization of cell selection action, and you can get the current selecting cell information to prevent the selection of a particular cell in a particular row. The [cellSelected](#) event will be triggered on completion of cell selection action, and you can get the current selected cell's information. The following code example demonstrates how to prevent the selection of the cell using the [cellSelecting](#) event.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',

```

```

        child: 'subtasks'
    },
    cellSelecting: function (args: any) {
        if (args.data.TaskID == 4 && args.cellIndex.cellIndex == 1) {
            args.cancel = true;
        }
    },
    selectionSettings: {
        mode: 'Cell'
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Filtering

Filtering in EJ2 JavaScript Gantt control

Filtering allows you to view specific or related records based on filter criteria. This can be done in the Gantt control by using the filter menu support and toolbar search support. To enable filtering in the Gantt control, set the [allowFiltering](#) to true. Menu filtering support can be configured using the [filterSettings](#) property and toolbar searching can be configured using the [searchSettings](#) property.

To use filter, inject the [Filter](#) module into the Gantt control.

Filter hierarchy modes

The Gantt supports a set of filtering modes with the [filterSettings.hierarchyMode](#) property. The following are the types of filter hierarchy modes available in the Gantt control:

- **Parent:** This is the default filter hierarchy mode in Gantt. The filtered records are displayed with its parent records. If the filtered records do not have any parent record, then only the filtered records will be displayed.
- **Child:** Displays the filtered records with its child record. If the filtered records do not have any child record, then only the filtered records will be displayed.
- **Both:** Displays the filtered records with its both parent and child records. If the filtered records do not have any parent and child records, then only the filtered records will be displayed.
- **None:** Displays only the filtered records.

INDEX.TS

```
import { Gantt, Filter } from '@syncfusion/ej2-gantt';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  allowFiltering: true
});
gantt.appendTo('#Gantt');
let dropDownMode: DropDownList = new DropDownList({
  dataSource: [
    { id: 'Parent', mode: 'Parent' },
    { id: 'Child', mode: 'Child' },
    { id: 'Both', mode: 'Both' },
    { id: 'None', mode: 'None' },
  ],
  fields: { text: 'mode', value: 'id' },
  value: 'Parent',
  change: (e: ChangeEventArgs) => {
    let mode: any = <string>e.value;
    gantt.filterSettings.hierarchyMode = mode;
    gantt.clearFiltering();
  }
});
dropDownMode.appendTo('#mode');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```



```

<title>EJ2 Gantt</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div style="padding-top: 7px; display: inline-block">Hierarchy
Mode</div>
            <div style="display: inline-block">
                <input type="text" id="mode">
            </div>
        </div>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Initial filter

To apply the filter at initial rendering, set the filter to `predicate` object in the [filterSettings.columns](#) property.

INDEX.TS

```

import { Gantt, Filter } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
    }
});

```

```

        child: 'subtasks'
      },
      filterSettings: {
        columns: [{ field: 'TaskName', matchCase: false, operator:
'startswith', predicate: 'and', value: 'Identify' },
        { field: 'TaskID', matchCase: false, operator: 'equal', predicate:
'and', value: 2 }]
      },
      allowFiltering: true
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter operators

The filter operator for a column can be defined in the `filterSettings.columns.operator` property.

The available operators and its supported data types are:

Operator | Description | Supported Types

startswith | Checks whether the value begins with the specified value. | String

endswith | Checks whether the value ends with the specified value. | String

contains | Checks whether the value contains the specified value. | String

equal | Checks whether the value is equal to the specified value. |String | Number | Boolean
| Date

notequal |Checks for the values that are not equal to the specified value. |String | Number
| Boolean | Date

greaterthan | Checks whether the value is greater than the specified value. | Number | Date

greaterthanorequal | Checks whether the value is greater than or equal to the specified value. | Number
| Date

lessthan | Checks whether the value is less than the specified value. | Number | Date

Function	Description	Input	Output
lessthanorequal	Checks whether the value is less than or equal to the specified value.	Number | Date	

By default, the `filterSettings.columns.operator` value is `equal`

Diacritics

By default, the Gantt control ignores the diacritic characters while filtering. To include diacritic characters, set the `filterSettings.ignoreAccent` to true.

In the following sample, type **Perform** in the **TaskName** column to filter diacritic characters.

INDEX.TS

```
import { Gantt, Filter } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            {TaskID: 2, TaskName: 'Identify site location', StartDate: new
Date('04/02/2019'), Duration: 0, Progress: 50 },
            {TaskID: 3, TaskName: 'Perform soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            {TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 0, Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            {TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50, resources:
[4]},
            {TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50},
            {TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0, Progress: 50 }
        ]
    }
]
```

```

    }]);
    Gantt.Inject(Filter);
    let gantt: Gantt = new Gantt({
        dataSource: GanttData,
        height: '450px',
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        },
        filterSettings: {
            ignoreAccent: true
        },
        allowFiltering: true
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filtering a specific column by method

You can filter the columns dynamically by using the [filterByColumn](#) method.

INDEX.TS

```
import { Gantt, Filter } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  allowFiltering: true
});
gantt.appendTo('#Gantt');
let filterBtn: Button = new Button();
filterBtn.appendTo('#filter');
document.getElementById('filter').addEventListener('click', () => {
  gantt.filterByColumn('TaskName', 'startswith', 'Iden');
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

    <button id="filter">Filter TaskName column</button>
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clear filtered columns

You can clear all the filtering condition done in the Gantt control by using the [clearFiltering](#) method.

The following code snippet explains the above behavior.

INDEX.TS

```

import { Gantt, Filter } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  allowFiltering: true,
  filterSettings: {
    columns: [{ field: 'TaskName', matchCase: false, operator:
'startswith', predicate: 'and', value: 'Identify' },
    { field: 'Progress', matchCase: false, operator: 'equal', predicate:
'and', value: 50 } ]
  }
});
gantt.appendTo('#Gantt');
let filterBtn: Button = new Button();
filterBtn.appendTo('#clearFilter');
document.getElementById('clearFilter').addEventListener('click', () => {
  gantt.clearFiltering();
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="clearFilter">Clear Filter</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter menu in EJ2 JavaScript Gantt control

The Gantt control provides the menu filtering support for each column. You can enable the filter menu by setting the [allowFiltering](#) to `true`. The filter menu UI will be rendered based on its column type, which allows you to filter data. You can filter the records with different operators.

INDEX.TS

```

import { Gantt, Filter } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    filterSettings: {
        type: 'Menu'
    },
    allowFiltering: true
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The [allowFiltering](#) property should be set to `true` to enable the filter menu.

Setting the [columns.allowFiltering](#) property to `false` prevents rendering filter menu for a particular column.

Custom component in filter menu

The [column.filter.ui](#) is used to add custom filter components to a particular column.

To implement custom filter ui, define the following functions:

- `create`: Creates custom component.
- `write`: Wire events for custom component.
- `read`: Read the filter value from custom component.

In the following sample, dropdown is used as custom component in the TaskName column.

INDEX.TS

```

import { Gantt, Filter } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { DataManager } from '@syncfusion/ej2-data';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { createElement } from '@syncfusion/ej2-base';
Gantt.Inject(Filter);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,

```



```

height: '450px',
taskFields: {
  id: 'TaskID',
  name: 'TaskName',
  startDate: 'StartDate',
  duration: 'Duration',
  progress: 'Progress',
  child: 'subtasks'
},
columns: [
  { field: 'TaskID' },
  { field: 'TaskName', filter: {
    ui: {
      create: (args: { target: Element, column: Object }) => {
        let db: Object = new
DataManager(gantt.treeGrid.grid.dataSource);
        let flValInput: HTMLElement = createElement('input',
{ className: 'flm-input' });
        args.target.appendChild(flValInput);
        this.dropInstance = new DropDownList({
          dataSource: new
DataManager(gantt.treeGrid.grid.dataSource),
          fields: { text: 'TaskName', value: 'TaskName' },
          placeholder: 'Select a value',
          popupHeight: '200px'
        });
        this.dropInstance.appendTo(flValInput);
      },
      write: (args: {
        column: Object, target: Element, parent: any,
        filteredValue: number | string
      }) => {
        this.dropInstance.value = args.filteredValue;
      },
      read: (args: { target: Element, column: any, operator:
string, fltrObj: Filter }) => {
        args.fltrObj.filterByColumn(args.column.field,
args.operator, this.dropInstance.value);
      }
    }
  }
],
allowFiltering: true
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Excel like filter in EJ2 JavaScript Gantt control

You can enable Excel like filter by defining [filterSettings.type](#) as `Excel`. The excel menu contains an option such as Sorting, Clear filter, Sub menu for advanced filtering.

INDEX.TS

```

import { Gantt, Filter } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  filterSettings: {
    type: 'Excel'
  },
  allowFiltering: true
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Searching in EJ2 JavaScript Gantt control

You can search records in the Gantt control by using the [search](#) method with search key as a parameter. The Gantt control provides an option to integrate the search text box in the toolbar by adding the search item to the [toolbar](#) property.

To search records, inject the **Filter** module into the Gantt control.

INDEX.TS

```

import { Gantt, Filter, Toolbar } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter, Toolbar);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  toolbar: ['Search']
});

```

```
});
ganttt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Initial search

In the Gantt control, you can load a task with some search criteria and this can be done by using the [searchSettings](#) property. To apply search at initial rendering, set the value for [fields](#), [operator](#), [key](#), and [ignoreCase](#) in the [searchSettings](#) property.

INDEX.TS

```
import { Gantt, Filter, Toolbar } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter, Toolbar);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
```

```

        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar: ['Search'],
    searchSettings: { fields: ['TaskName'], operator: 'contains', key:
    'List', ignoreCase: true }
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, Gantt searches all the bound column values. To customize this behavior, define the [searchSettings.fields](#) property.

Search operators

The search operator can be defined in the [searchSettings.operator](#) property to configure specific searching.

The following operators are supported in searching:

Operator | Description

startsWith | Checks whether a value begins with the specified value.

endsWith | Checks whether a value ends with the specified value.

contains | Checks whether a value contains the specified value.

equal | Checks whether a value is equal to the specified value.

notEqual | Checks for the values that are not equal to the specified value.

By default, the [searchSettings.operator](#) value is **contains**.

Search by external button

To search the Gantt records from an external button, invoke the [search](#) method.

INDEX.TS

```
import { Gantt, Filter, Toolbar } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter, Toolbar);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  allowFiltering: true
});
gantt.appendTo('#Gantt');
let searchBtn: Button = new Button();
searchBtn.appendTo('#search');
document.getElementById('search').addEventListener('click', () => {
  let searchText: string =
    (<HTMLInputElement>document.getElementsByClassName('searchtext')[0]).value;
  gantt.search(searchText);
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-float-input" style="width: 200px; display:
inline-block;">
            <input type="text" class="searchtext">
            <span class="e-float-line"></span>
            <label class="e-float-text">Search text</label>
        </div>
        <button id="search">Search</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: You should set the [allowFiltering](#) property to `true` for searching the content externally.

Search specific columns

By default, the Gantt control searches all the columns. You can search specific columns by defining the specific column's field names in the [searchSettings.fields](#) property.

INDEX.TS

```

import { Gantt, Filter, Toolbar } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter, Toolbar);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
        { field: 'TaskName', headerText: 'Task Name', width: '250' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar: ['Search'],

```

```

        searchSettings: { fields: ['TaskName', 'Duration'] }
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

In above sample, you can search only **TaskName** and **Duration** column values.

Clear search by external button

You can set [searchSettings.key](#) property as **empty** string, to clear the searched Gantt records from external button.

INDEX.TS

```

import { Gantt, Filter, Toolbar } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter, Toolbar);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },

```



```

        { field: 'TaskName', headerText: 'Task Name', width: '250' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar: ['Search'],
    searchSettings: { fields: ['TaskName'], operator: 'contains', key:
'Perform', ignoreCase: true },
});
gantt.appendTo('#Gantt');
let clearBtn: Button = new Button();
clearBtn.appendTo('#clear');
document.getElementById('clear').addEventListener('click', () => {
    gantt.searchSettings.key='';
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="clear">Clear Search</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Task dependency in EJ2 JavaScript Gantt control

Task dependency or task relationship can be established between two tasks in Gantt. This dependency affects the project schedule. If you change the predecessor of a task, it will affect the successor task, which will affect the next task, and so on. Relationship can be established between parent-parent tasks, child-child tasks, parent-child and child-parent task.

In Gantt, you can enable or disable the parent predecessor using [allowParentDependency](#) property.

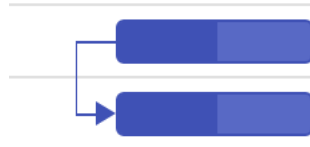
By default, the `allowParentDependency` property will be `true`.

Task relationship types

Task relationships are categorized into four types based on the start and finish dates of the task.

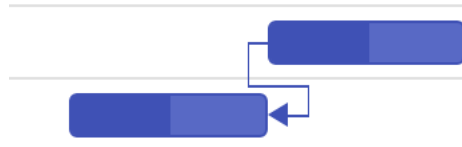
Start to Start (SS)

You cannot start a task until the dependent task also starts.



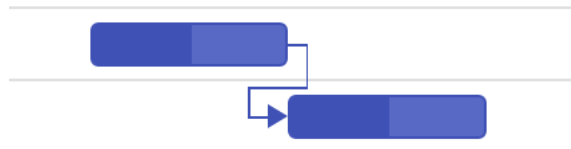
Start to Finish (SF)

You cannot finish a task until the dependent task is started.



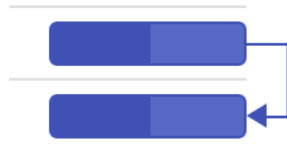
Finish to Start (FS)

You cannot start a task until the dependent task is completed.



Finish to Finish (FF)

You cannot finish a task until the dependent task is completed.



Define task relationship

Task relationship is defined in the data source as a string value, and this value is mapped to the Gantt control by using the [taskFields.dependency](#) property. The following code example demonstrates how to enable the predecessor in the Gantt control.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4,Predecessor:"2FS", Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0,Predecessor:"6SS", Progress: 50 }
        ]
    },
];
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        dependency: 'Predecessor',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Predecessor offset with duration units

In the Gantt control, the predecessor offset can be defined with the following duration units:

- Day
- Hour
- Minute

You can define an offset with various offset duration units for predecessors by using the following code example.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
      { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
      { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/04/2019'), Duration: 4, Predecessor: "2FS+2days", Progress: 50 },
      { TaskID: 5, TaskName: 'Clear the building site', StartDate:
new Date('04/04/2019'), Duration: 2, Progress: 30, Predecessor: '4FF+960m'}
    ]
  }
]

```

```

    },
    {
        TaskID: 6,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 7, TaskName: 'Develop floor plan for estimation',
              StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'List materials', StartDate: new
              Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 9, TaskName: 'Estimation approval', StartDate: new
              Date('04/06/2019'), Duration: 0, Predecessor: "7SS+16h", Progress: 50 }
        ]
    },
    ],
    let gantt: Gantt = new Gantt({
        dataSource: GanttData,
        height: '450px',
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            dependency: 'Predecessor',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        },
        columns: [
            { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
              '100' },
            { field: 'Predecessor', headerText: 'Depedency', width: '150' },
            { field: 'TaskName', headerText: 'Task Name', width: '150' },
            { field: 'StartDate', headerText: 'Start Date', width: '150' },
            { field: 'Duration', headerText: 'Duration', width: '150' },
            { field: 'Progress', headerText: 'Progress', width: '150' }
        ]
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
    rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disabling automatic dependency offset updates

By default, the dependency offsets are automatically updated in the Gantt chart whenever a task's start or end date is changed. However, if you want to disable this feature, you can do so by disabling the [updateOffsetOnTaskbarEdit](#) property. Once this property is disabled, you can only update the offset value by editing the predecessor column cell or the offset column in the dependency tab of the edit dialog.

INDEX.TS

```

import { Gantt, Edit, Toolbar } from '@syncfusion/ej2-gantt';
Gantt.Inject(Edit, Toolbar);
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/04/2019'), Duration: 4, Predecessor: "2FS+2days", Progress: 50 },
            { TaskID: 5, TaskName: 'Clear the building site', StartDate:
new Date('04/04/2019'), Duration: 2, Progress: 30, Predecessor: '4FF+960m'}
        ]
    },
    {
        TaskID: 6,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 7, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },

```

```

        { TaskID: 8, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
        { TaskID: 9, TaskName: 'Estimation approval', StartDate: new
Date('04/06/2019'), Duration: 0,Predecessor:"7SS+16h", Progress: 50 }
    ]
    },
    ];
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        dependency: 'Predecessor',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    updateOffsetOnTaskbarEdit :false,
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    toolbar: ['Add', 'Cancel', 'CollapseAll', 'Delete', 'Edit',
'ExpandAll', 'NextTimeSpan', 'PrevTimeSpan', 'Search', 'Update', 'Indent',
'Outdent']
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
        { field: 'Predecessor', headerText: 'Depedency', width: '150' },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' }
    ]
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Validate predecessor links on editing

In Gantt, task relationship link can be broken by editing the start date, end date and duration value of task. When the task relationship broken on any edit action. This can be handled in Gantt in two ways.

- Using actionBegin event
- Using predecessor validation dialog

Using actionBegin event

When the task relationship link is broken on any edit action, then the [actionBegin](#) event will be triggered with `requestType` argument as `validateLinkedTask`. You can validate the editing action within the actionBegin event using the `validateMode` event argument. The `validateMode` event argument has the following properties:

Argument | Default value | Description

`args.validateMode.respectLink` | false | In this validation mode, the predecessor links get high priority. With this mode enabled, when the successor task is moved before the predecessor task's end date, the editing will be reverted, and dates will be validated based on the dependency links.

`args.validateMode.removeLink` | false | In this validation mode, the taskbar editing gets high priority. For inappropriate task dates, the dependency links will be removed and tasks will be moved to the edited date.

`args.validateMode.preserveLinkWithEditing` | true | In this validation mode, the taskbar editing will be considered along with the dependency links. This relationship will be maintained by updating the offset value of predecessors.

By default, the `preserveLinkWithEditing` validation mode will be enabled, so the predecessors are updated with offset values.

The following sample explains enabling the `respectLink` validation mode while editing the linked tasks in the [actionBegin](#) event.

INDEX.TS

```

import { Gantt, Edit } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4,Predecessor:"2FS", Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0,Predecessor:"6SS", Progress: 50 }
        ]
    },
];
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        dependency: 'Predecessor',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowTaskbarEditing: true
    },
    actionBegin: (args: any) => {
        if (args.requestType == "validateLinkedTask") {
            args.validateMode.respectLink = true;
        }
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Using validation dialog

When disabling all the validation modes in the [actionBegin](#) event, a validation pop-up will be displayed prompting users to select the validation mode to validate taskbar editing.

This validation pop-up will display different options based on the successor task's start date after editing.

If you move the successor task that starts after the predecessor task's end date, then a dialog will be rendered with the following options:

- Cancel, Keep the existing link.
- Remove the link and move the task to start on edited date.
- Move the task to start on edited date and keep the link.

If you move the successor task that starts before the predecessor task's end date, then a dialog will be rendered with the following options:

- Cancel, Keep the existing link.
- Remove the link and move the task to start on edited date.

The following code example shows how to enable the predecessor validation dialog in Gantt.

INDEX.TS

```
import { Gantt, Edit } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4,Predecessor:"2FS", Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0,Predecessor:"6SS", Progress: 50 }
        ]
    },
];
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        dependency: 'Predecessor',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowTaskbarEditing: true
    },
    actionBegin: (args: any) => {
        if (args.requestType == "validateLinkedTask") {
            args.validateMode.preserveLinkWithEditing = false;
        }
    }
});
```

```
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Dynamically show/hide the dependency line

By default, mapping the dependency field in taskFields displays dependency lines in the Gantt chart. To hide the dependency line upon button click, set **visibility** style to hidden for the CSS class name **.e-gantt-dependency-view-container**.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { Switch } from '@syncfusion/ej2-buttons';
let GanttData: Object[] = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      { TaskID: 2, TaskName: 'Identify Site location', StartDate: new
Date('04/02/2019'), Duration: 0, Progress: 50, },
    ]
  }
];
```

```

    { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, },
    { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/04/2019'), Duration: 4, Predecessor: '2FS+2days', Progress: 50, },
    { TaskID: 5, TaskName: 'Clear the building site', StartDate: new
Date('04/04/2019'), Duration: 2, Progress: 30, Predecessor: '4FF+960m', },
  ],
},
{
  TaskID: 6,
  TaskName: 'Project Estimation',
  StartDate: new Date('04/02/2019'),
  EndDate: new Date('04/21/2019'),
  subtasks: [
    { TaskID: 7, TaskName: 'Develop floor plan for estimation', StartDate:
new Date('04/04/2019'), Duration: 3, Progress: 50, },
    { TaskID: 8, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, },
    { TaskID: 9, TaskName: 'Estimation approval', StartDate: new
Date('04/06/2019'), Duration: 0, Predecessor: '7SS+16h',Progress: 50, },
  ],
},
];
let switchObj: Switch = new Switch({ checked: false, change: Onchange });
switchObj.appendTo('#switch');
function Onchange() {
  let ganttDependencyViewContainer = document.querySelector('.e-gantt-
dependency-view-container') as HTMLElement;
  if (switchObj.checked) {
    ganttDependencyViewContainer.style.visibility = 'hidden';
  } else {
    ganttDependencyViewContainer.style.visibility = 'visible';
  }
}
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    dependency: 'Predecessor',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
    { field: 'Predecessor', headerText: 'Depedency', width: '150' },
    { field: 'TaskName', headerText: 'Task Name', width: '150' },
    { field: 'StartDate', headerText: 'Start Date', width: '150' },
    { field: 'Duration', headerText: 'Duration', width: '150' },
    { field: 'Progress', headerText: 'Progress', width: '150' },
  ],
});
gantt.appendTo('#Default');
```

INDEX.HTML

```

<html>
  <head>
    <script
src="https://ej2.syncfusion.com/javascript/demos/gantt/default/datasource.js
" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet"/>
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet"/>
    <style> body { touch-action: none; } </style>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
  </head>
  <body>
    <div class="stackblitz-container material3">
      <div class="control-section">
        <div class="content-wrapper">
          <label>Show/Hide Dependency Line</label>
          <input type="checkbox" id="switch" />
          <div id="Default"></div>
        </div>
      </div>
    </div>
    <script src="index.js" type="text/javascript"></script>
  </body>
</html>

```

Sorting in EJ2 JavaScript Gantt control

Sorting enables you to sort data in the ascending or descending order. To sort a column, click the column header.

To sort multiple columns, press and hold the CTRL key and click the column header. You can clear sorting of any one of the multi-sorted columns by pressing and holding the SHIFT key and clicking the specific column header.

To enable sorting in the Gantt control, set the [allowSorting](#) property to true. Sorting options can be configured through the [sortSettings](#) property.

To sort, inject the [Sort](#) module into the Gantt control.

INDEX.TS

```

import { Gantt, Sort } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Sort);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  allowSorting: true,
  taskFields: {

```

```

        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    }
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* Gantt columns are sorted in the ascending order. If you click the already sorted column, the sort direction toggles.

* To disable sorting for a particular column, set the [columns.allowSorting](#) property to false.

Sorting column on Gantt initialization

The Gantt control can be rendered with sorted columns initially, and this can be achieved by using the [sortSettings](#) property. You can add columns that are sorted initially in the [sortSettings.columns](#) collection defined with [field](#) and [direction](#) properties. The following code example shows how to add the sorted column to Gantt initialization.

INDEX.TS

```
import { Gantt, Sort } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Sort);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  sortSettings: { columns: [{ field: 'TaskID', direction: 'Ascending' }, {
    field: 'TaskName', direction: 'Ascending' }] },
  allowSorting: true
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```


Sorting column dynamically

Columns in the Gantt control can be sorted dynamically using the [sortColumn](#) method. The following code example demonstrates how to invoke the [sortColumn](#) method by clicking the custom button.

INDEX.TS

```
import { Gantt, Sort } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Sort);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  allowSorting: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  }
});
gantt.appendTo('#Gantt');
let sortBtn: Button = new Button();
sortBtn.appendTo('#sortColumn');
document.getElementById('sortColumn').addEventListener('click', () => {
  gantt.sortModule.sortColumn('TaskName', "Descending", false)
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <button id="sortColumn">Sort TaskName Column</button>

  <div id="container">
    <div id="Gantt"></div>
  </div>
</body>
</html>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clear all the sorted columns dynamically

In the Gantt control, you can clear all the sorted columns and return to previous position using the [clearSorting](#) public method. The following code snippet shows how to clear all the sorted columns by clicking the custom button.

INDEX.TS

```

import { Gantt, Sort } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Sort);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    sortSettings: { columns: [{ field: 'TaskID', direction: 'Ascending' }, {
field: 'TaskName', direction: 'Ascending' }] },
    allowSorting: true
});
gantt.appendTo('#Gantt');
let clrBtn: Button = new Button();
clrBtn.appendTo('#clearSorting');
document.getElementById('clearSorting').addEventListener('click', () => {
    gantt.clearSorting();
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="clearSorting">Clear Sorting</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sorting events

During the sort action, the Gantt control triggers two events. The [actionBegin](#) event triggers before the sort action starts, and the [actionComplete](#) event triggers after the sort action is completed.

INDEX.TS

```

import { Gantt, Sort } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { SortEventArgs } from '@syncfusion/ej2-grids';
Gantt.Inject(Sort);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowSorting: true,
    actionBegin: actionHandler,
    actionComplete: actionHandler
});
gantt.appendTo('#Gantt');
function actionHandler(args: SortEventArgs) {
    alert(args.requestType + ' ' + args.type);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The `args.requestType` is the current action name. For example, for sorting the `args.requestType`, value is **sorting**.

Sorting Custom Columns

In Gantt, you can sort custom columns of different types like string, numeric, etc., By adding the custom column in the column collection,

you can perform initial sort using the `sortSettings` or you can also sort the column dynamically by a button click.

The following code snippets explains how to achieve this.

INDEX.TS

```

import { Gantt, Sort } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
Gantt.Inject(Sort);
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [

```

```

        { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50, /*CustomColumn:
'BCustomColumn'*/ CustomColumn: '2' },
        { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, /*CustomColumn:
'BCustomColumn'*/ CustomColumn: '3' },
        { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, /*CustomColumn:
'BCustomColumn'*/ CustomColumn: '4' },
    ]
},
{
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
        { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50,
/*CustomColumn: 'BCustomColumn'*/ CustomColumn: '6' },
        { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, /*CustomColumn:
'BCustomColumn'*/ CustomColumn: '1' },
        { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, /*CustomColumn:
'BCustomColumn'*/ CustomColumn: '5' }
    ]
},
];
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    allowSorting: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        { field: 'Progress', headerText: 'Progress' },
        { field: 'TaskName', headerText: 'Task Name' },
        { field: 'StartDate', headerText: 'Start Date' },
        { field: 'Duration', headerText: 'Duration' },
        { field: 'CustomColumn', headerText: 'CustomColumn' }
    ]
});
gantt.appendTo('#Gantt');
let sortBtn: Button = new Button();
sortBtn.appendTo('#sortColumn');
document.getElementById('sortColumn').addEventListener('click', () => {
    gantt.sortModule.sortColumn('CustomColumn', "Ascending", false)
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <button id="sortColumn">Sort Custom Column</button>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Baseline in EJ2 JavaScript Gantt control

The baseline feature enables users to view the deviation between the planned dates and actual dates of the tasks in a project. Baseline dates or planned dates of a task may or may not be same as the actual task dates. The baseline can be enabled by setting the [renderBaseline](#) property to true and the baseline color can be changed using the [baselineColor](#) property. To render the baseline, you should map the baseline start and end date values from the data source. This can be done using the [baselineStartDate](#) and [baselineEndDate](#) properties. The following code example shows how to enable a baseline in the Gantt control.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [

```

```

        { TaskID: 2, TaskName: 'Identify Site location',
BaselineStartDate: new Date('04/02/2019'), BaselineEndDate: new
Date('04/06/2019'), StartDate: new Date('04/02/2019'), Duration: 0,
Progress: 50 },
        { TaskID: 3, TaskName: 'Perform Soil test', BaselineStartDate:
new Date('04/04/2019'), BaselineEndDate: new Date('04/09/2019'), StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
        { TaskID: 4, TaskName: 'Soil test approval', BaselineStartDate:
new Date('04/08/2019'), BaselineEndDate: new Date('04/12/2019'), StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
    ]
},
{
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
        { TaskID: 6, TaskName: 'Develop floor plan for estimation',
BaselineStartDate: new Date('04/04/2019'), BaselineEndDate: new
Date('04/08/2019'), StartDate: new Date('04/04/2019'), Duration: 3,
Progress: 50 },
        { TaskID: 7, TaskName: 'List materials', BaselineStartDate: new
Date('04/02/2019'), BaselineEndDate: new Date('04/04/2019'), StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
        { TaskID: 8, TaskName: 'Estimation approval', BaselineStartDate:
new Date('04/02/2019'), BaselineEndDate: new Date('04/02/2019'), StartDate:
new Date('04/04/2019'), Duration: 0, Progress: 50 }
    ]
},
];
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        baselineStartDate: "BaselineStartDate",
        baselineEndDate: "BaselineEndDate",
        child: 'subtasks'
    },
    renderBaseline: true,
    baselineColor: 'red'
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Timeline

Timeline in EJ2 JavaScript Gantt control

In the Gantt control, timeline is used to represent the project duration as individual cells with defined unit and formats.

Timeline view modes

Gantt contains the following in-built timeline view modes:

- Hour
- Day
- Week
- Month
- Year

Timescale mode in Gantt can be defined by using [timelineViewMode](#) property and also we can define timescale mode of top tier and bottom tier by using [topTier.unit](#) and [bottomTier.unit](#) properties.

Week timeline mode

In the **Week** timeline mode, the upper part of the schedule header displays the weeks, whereas the bottom half of the header displays the days. Refer to the following code example.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { WeekData } from 'datasource.ts';
let gantt: Gantt = new Gantt({

```



```

dataSource: WeekData,
height: '450px',
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
},
timelineSettings: {
    timelineViewMode: 'Week'
}
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Month timeline mode

In the **Month** timeline mode, the upper part of the schedule header displays the months, whereas the bottom header of the schedule displays its corresponding weeks. Refer to the following code example.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { MonthData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: MonthData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    timelineSettings: {
        timelineViewMode: 'Month',
        timelineUnitSize: 150
    }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Year timeline mode

In the **Year** timeline mode, the upper schedule header displays the years whereas, the bottom header displays its corresponding months. Refer to the following code example.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { YearData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: YearData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  timelineSettings: {
    timelineViewMode: 'Year',
    timelineUnitSize: 70
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Day timeline mode

In the **Day** timeline mode, the upper part of the header displays the days whereas, the bottom schedule header displays its corresponding hours. Refer to the following code example.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { DayData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: DayData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    timelineSettings: {
        timelineViewMode: 'Day'
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hour timeline mode

An **Hour** timeline mode tracks the tasks in minutes scale. In this mode, the upper schedule header displays hour scale and the lower schedule header displays its corresponding minutes.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { HourData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: HourData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    timelineSettings: {
        timelineViewMode: 'Hour'
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Week start day customization

In the Gantt control, you can customize the week start day using the [weekStartDay](#) property. By default, the [weekStartDay](#) is set to 0, which specifies the Sunday as a start day of the week. But, you can customize the week start day by using the following code example.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { WeekStartData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: WeekStartData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  timelineSettings: {
    timelineViewMode: 'Week',
    weekStartDay: 3
  }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize automatic timescale update action

In the Gantt control, the schedule timeline will be automatically updated when the tasks date values are updated beyond the project start date and end date ranges. This can be enabled or disabled using the [updateTimescaleView](#) property.

INDEX.TS

```

import { Gantt, Edit } from '@syncfusion/ej2-gantt';
import { timelineData } from 'datasource.ts';
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
    dataSource: timelineData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    timelineSettings: {
        updateTimescaleView: false
    },
    editSettings: {
        allowEditing: true,
        allowTaskbarEditing: true
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Top tier and bottom tier in EJ2 JavaScript Gantt control

Gantt control contains two tier layout in timeline, we can customize the top tier and bottom tier using [topTier](#) and [bottomTier](#) properties. Timeline tier's unit can be defined by using [unit](#) property and [format](#) property was used to define date format of timeline cell and [count](#) property was used to define how many unit will be combined as single cell and [formatter](#) property was used to define custom method to format the date value of timeline cell.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  timelineSettings: {
    topTier: {
      format: 'MMM',
      unit: 'Month'
    },
  },
});

```



```

        bottomTier: {
            unit: 'Day'
        }
    }
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Combining timeline cells

In Gantt, timeline cells in top tier and bottom tier can be combined with number of timeline units, this can be achieved by using [topTier.count](#) and [bottomTier.count](#) properties. Please refer the below sample.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { Data } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: Data,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',

```

```

        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    timelineSettings: {
        timelineUnitSize: 100,
        topTier: {
            unit: 'Year'
        },
        bottomTier: {
            unit: 'Month',
            count: 6
        }
    }
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Format value of timeline cell

In the Gantt control, you can format the value of top and bottom timeline cells using the standard date format string or the custom formatter method. This can be done using the [topTier.format](#),

[topTier.formatter](#), [bottomTier.format](#) and [bottomTier.formatter](#) properties. The following example shows how to use the formatter method for timeline cells.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { Data } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: Data,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  timelineSettings: {
    topTier: {
      unit: 'Month',
      count: 3,
      formatter: (date) => {
        var month = date.getMonth();
        if (month >= 0 && month <= 2) {
          return 'Q1';
        } else if (month >= 3 && month <= 5) {
          return 'Q2';
        } else if (month >= 6 && month <= 8) {
          return 'Q3';
        } else {
          return 'Q4';
        }
      }
    },
    bottomTier: {
      unit: 'Month',
      format: 'MMM'
    }
  },
  projectStartDate: new Date('01/04/2019'),
  projectEndDate: new Date('12/30/2019')
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Timeline cell width

In the Gantt control, you can define the width value of timeline cell using the [timelineSettings.timelineUnitSize](#) property. This value will be set to the bottom timeline cell, and the width value of top timeline cell will be calculated automatically based on bottom tier cell width using the [topTier.unit](#) and [timelineSettings.timelineUnitSize](#) properties. Refer to the following example.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    timelineSettings: {
        timelineUnitSize: 200
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Zooming in EJ2 JavaScript Gantt control

The zooming support provides options to increase or decrease the width of timeline cells and also provides options to change the timeline units dynamically. This support enables you to view the tasks in a project clearly from minute to decade timespan. To enable the zooming features, define the **ZoomIn**, **ZoomOut**, and **ZoomToFit** items to toolbar items collections, and this action can be performed on external actions such as button click using the [zoomIn](#), [zoomOut](#), and [fitToProject](#) built-in methods. The following zooming options are available to view the project:

Zoom in

This support is used to increase the timeline width and timeline unit from years to minutes timespan. When the **ZoomIn** icon was clicked, the timeline cell width is increased when the cell size exceeds the specified range and the timeline unit is changed based on the current zoom levels.

Zoom out

This support is used to increase the timeline width and timeline unit from minutes to years timespan. When the **ZoomOut** icon was clicked, the timeline cell width is decreased when the cell size falls behind the specified range and the timeline view mode is changed based on the current zooming levels.

Zoom to fit

This support is used to view all the tasks available in a project within available area on the chart part of Gantt. When users click the **ZoomToFit** icon, then all the tasks are rendered within the available chart container width.

INDEX.TS

```
import { Gantt, Toolbar } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  toolbar: ['ZoomIn', 'ZoomOut', 'ZoomToFit'],
  labelSettings: {
    leftLabel: 'TaskName'
  },
  projectStartDate: new Date('03/24/2019'),
  projectEndDate: new Date('04/28/2019')
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

Customizing zooming levels

In Gantt, the zoom in and zoom out actions are performed based on the predefined zooming levels in the **zoomingLevels** property. You can customize the zooming actions by defining the required zooming collection to the **zoomingLevels** property.

INDEX.TS

```
import { Gantt, Toolbar, ZoomTimelineSettings } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let customZoomingLevels: ZoomTimelineSettings[] = [{
    topTier: { unit: 'Month', format: 'MMM, yy', count: 1 },
    bottomTier: { unit: 'Week', format: 'dd', count: 1 }, timelineUnitSize:
33, level: 0,
    timelineViewMode: 'Month', weekStartDay: 0, updateTimescaleView: true,
weekendBackground: null, showTooltip: true
},
{
    topTier: { unit: 'Month', format: 'MMM, yyyy', count: 1 },
    bottomTier: { unit: 'Week', format: 'dd MMM', count: 1 },
timelineUnitSize: 66, level: 1,
    timelineViewMode: 'Month', weekStartDay: 0, updateTimescaleView: true,
weekendBackground: null, showTooltip: true
},
{
    topTier: { unit: 'Month', format: 'MMM, yyyy', count: 1 },
    bottomTier: { unit: 'Week', format: 'dd MMM', count: 1 },
timelineUnitSize: 99, level: 2,
    timelineViewMode: 'Month', weekStartDay: 0, updateTimescaleView: true,
weekendBackground: null, showTooltip: true
},
{
    topTier: { unit: 'Week', format: 'MMM dd, yyyy', count: 1 },
    bottomTier: { unit: 'Day', format: 'd', count: 1 }, timelineUnitSize:
33, level: 3,
    timelineViewMode: 'Week', weekStartDay: 0, updateTimescaleView: true,
weekendBackground: null, showTooltip: true
},
{
    topTier: { unit: 'Week', format: 'MMM dd, yyyy', count: 1 },
    bottomTier: { unit: 'Day', format: 'd', count: 1 }, timelineUnitSize:
66, level: 4,
    timelineViewMode: 'Week', weekStartDay: 0, updateTimescaleView: true,
weekendBackground: null, showTooltip: true
},
{
    topTier: { unit: 'Day', format: 'E dd yyyy', count: 1 },
    bottomTier: { unit: 'Hour', format: 'hh a', count: 12 },
timelineUnitSize: 66, level: 5,
    timelineViewMode: 'Day', weekStartDay: 0, updateTimescaleView: true,
weekendBackground: null, showTooltip: true
},
{
    topTier: { unit: 'Day', format: 'E dd yyyy', count: 1 },
```

```

        bottomTier: { unit: 'Hour', format: 'hh a', count: 6 },
        timelineUnitSize: 99, level: 6,
        timelineViewMode: 'Day', weekStartDay: 0, updateTimescaleView: true,
        weekendBackground: null, showTooltip: true
    },
    ];
    Gantt.Inject(Toolbar);
    let gantt: Gantt = new Gantt({
        dataSource: GanttData,
        height: '450px',
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            dependency: 'Predecessor',
            child: 'subtasks'
        },
        toolbar: ['ZoomIn', 'ZoomOut', 'ZoomToFit'],
        labelSettings: {
            leftLabel: 'TaskName'
        },
        dataBound: function () {
            gantt.zoomingLevels = customZoomingLevels;
        },
        projectStartDate: new Date('03/24/2019'),
        projectEndDate: new Date('04/28/2019')
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
    rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

```



```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Zoom action by methods

You can perform the various zoom actions dynamically or on external click action using the following methods:

- Zoom in - [zoomIn](#)
- Zoom out - [zoomOut](#)
- Fit to project - [fitToProject](#)

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  labelSettings: {
    leftLabel: 'TaskName'
  },
  projectStartDate: new Date('03/24/2019'),
  projectEndDate: new Date('04/28/2019')
});
gantt.appendTo('#Gantt');
let zoomInBtn: Button = new Button();
zoomInBtn.appendTo('#zoomIn');
document.getElementById('zoomIn').addEventListener('click', () => {
  gantt.zoomIn();
});
let zoomOutBtn: Button = new Button();
zoomOutBtn.appendTo('#zoomOut');
document.getElementById('zoomOut').addEventListener('click', () => {
  gantt.zoomOut();
});
let fitToBth: Button = new Button();
fitToBth.appendTo('#fitToProject');
document.getElementById('fitToProject').addEventListener('click', () => {
  gantt.fitToProject();
});
```

```
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="zoomIn">ZoomIn</button>
    <button id="zoomOut">ZoomOut</button>
    <button id="fitToProject">FitToProject</button>
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Timezone in EJ2 JavaScript Gantt control

The Gantt uses the current system time zone by default. If it needs to follow some other user-specified time zone, then the `timezone` property must be used.

Understanding date manipulation in JavaScript

The `new Date()` in JavaScript returns the exact current date object with complete time and timezone information. For example, it may return a value such as `Wed Dec 12, 2018, 05:23:27 GMT+0530 (India Standard Time)` which indicates that the current date is December 12, 2018, and the current time is 5.23 AM on browsers following the IST timezone.

Display same time everywhere with no time difference

Setting `timezone` to UTC for Gantt will display the same time as in the database for all the users in different time zone.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt(
{
    dataSource: [
        { taskID: 1, taskName: 'Project Schedule', startDate: new
Date('02/04/2019 08:00'), endDate: new Date('03/10/2019') },
        { taskID: 2, taskName: 'Planning', startDate: new Date('02/04/2019
08:00'), endDate: new Date('02/10/2019'), parentID: 1 },
        { taskID: 3, taskName: 'Plan timeline', startDate: new
Date('02/04/2019 08:00'), endDate: new Date('02/10/2019'), duration: 6,
progress: '60', parentID: 2 },
        { taskID: 4, taskName: 'Plan budget', startDate: new Date('02/04/2019
08:00'), endDate: new Date('02/10/2019'), duration: 6, progress: '90',
parentID: 2 },
        { taskID: 5, taskName: 'Allocate resources', startDate: new
Date('02/04/2019 08:00'), endDate: new Date('02/10/2019'), duration: 6,
progress: '75', parentID: 2 },
        { taskID: 6, taskName: 'Planning complete', startDate: new
Date('02/06/2019 08:00'), endDate: new Date('02/10/2019'), duration: 0,
predecessor: '3FS,4FS,5FS', parentID: 2 },
        { taskID: 7, taskName: 'Design', startDate: new Date('02/13/2019
08:00'), endDate: new Date('02/17/2019 08:00'), parentID: 1 },
        { taskID: 8, taskName: 'Software Specification', startDate: new
Date('02/13/2019 08:00'), endDate: new Date('02/15/2019'), duration: 3,
progress: '60', predecessor: '6FS', parentID: 7 },
        { taskID: 9, taskName: 'Develop prototype', startDate: new
Date('02/13/2019 08:00'), endDate: new Date('02/15/2019'), duration: 3,
progress: '100', predecessor: '6FS', parentID: 7 },
        { taskID: 10, taskName: 'Get approval from customer', startDate: new
Date('02/16/2019 08:00'), endDate: new Date('02/17/2019 08:00'), duration:
2, progress: '100', predecessor: '9FS', parentID: 7 },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/17/2019 08:00'), endDate: new Date('02/17/2019 08:00'), duration:
0, predecessor: '10FS', parentID: 7 }
    ],
    taskFields: {
        id: 'taskID',
        name: 'taskName',
        startDate: 'startDate',
        duration: 'duration',
        progress: 'progress',
        dependency: 'predecessor',
        parentID: 'parentID'
    },
    timelineSettings: {
        timelineUnitSize: 65,
        topTier: {
            unit: 'Day',
            format: 'MMM dd, yyyy'
        },
        bottomTier: {
            unit: 'Hour',
            format: 'hh:mm a'
        }
    },
    timezone: 'UTC',

```

```

        durationUnit: 'Hour',
        includeWeekend: true,
        dateFormat: 'hh:mm a',
        height: '450px',
        dayWorkingTime: [{ from: 0, to: 23 }],
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

CRUD operations with timezone

CRUD operations can be performed with timezone and the Gantt is rendered based on the timezone specified in the load time. All the editing actions will be done based on the user timezone, but on database save action, we have reversed this conversion to local time and provided data to client-side events for better understanding. Refer to the following code example.

INDEX.TS

```

import { Gantt, Edit, Selection } from '@syncfusion/ej2-gantt';
Gantt.Inject(Edit, Selection);
let GanttData: Object[] = [
  {
    taskID: 1,
    taskName: 'Project Schedule',

```

```

    startDate: new Date('02/04/2019 08:00'),
    endDate: new Date('03/10/2019')
  },
  {
    taskID: 2,
    taskName: 'Planning',
    startDate: new Date('02/04/2019 08:00'),
    endDate: new Date('02/10/2019'),
    parentID: 1
  },
  {
    taskID: 3,
    taskName: 'Plan timeline',
    startDate: new Date('02/04/2019 08:00'),
    endDate: new Date('02/10/2019'),
    duration: 6,
    progress: '60',
    parentID: 2
  },
  {
    taskID: 4,
    taskName: 'Plan budget',
    startDate: new Date('02/04/2019 08:00'),
    endDate: new Date('02/10/2019'),
    duration: 6,
    progress: '90',
    parentID: 2
  },
  {
    taskID: 5,
    taskName: 'Allocate resources',
    startDate: new Date('02/04/2019 08:00'),
    endDate: new Date('02/10/2019'),
    duration: 6,
    progress: '75',
    parentID: 2
  },
  {
    taskID: 6,
    taskName: 'Planning complete',
    startDate: new Date('02/06/2019 08:00'),
    endDate: new Date('02/10/2019'),
    duration: 0,
    predecessor: '3FS,4FS,5FS',
    parentID: 2
  },
  {
    taskID: 7,
    taskName: 'Design',
    startDate: new Date('02/13/2019 08:00'),
    endDate: new Date('02/17/2019 08:00'),
    parentID: 1
  },
  {
    taskID: 8,
    taskName: 'Software Specification',
    startDate: new Date('02/13/2019 08:00'),

```

```

        endDate: new Date('02/15/2019'),
        duration: 3,
        progress: '60',
        predecessor: '6FS',
        parentID: 7
    },
    {
        taskID: 9,
        taskName: 'Develop prototype',
        startDate: new Date('02/13/2019 08:00'),
        endDate: new Date('02/15/2019'),
        duration: 3,
        progress: '100',
        predecessor: '6FS',
        parentID: 7
    },
    {
        taskID: 10,
        taskName: 'Get approval from customer',
        startDate: new Date('02/16/2019 08:00'),
        endDate: new Date('02/17/2019 08:00'),
        duration: 2,
        progress: '100',
        predecessor: '9FS',
        parentID: 7
    },
    {
        taskID: 11,
        taskName: 'Design complete',
        startDate: new Date('02/17/2019 08:00'),
        endDate: new Date('02/17/2019 08:00'),
        duration: 0,
        predecessor: '10FS',
        parentID: 7
    }
];
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    editSettings: {
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    taskFields: {
        id: 'taskID',
        name: 'taskName',
        startDate: 'startDate',
        duration: 'duration',
        progress: 'progress',
        dependency: 'predecessor',
        parentID: 'parentID'
    },
    actionComplete: actionComplete,
    timelineSettings: {
        timelineUnitSize: 65,
        topTier: {

```

```

        unit: 'Day',
        format: 'MMM dd, yyyy'
    },
    bottomTier: {
        unit: 'Hour',
        format: 'hh:mm a'
    }
},
timezone: 'America/New_York',
durationUnit: 'Hour',
includeWeekend: true,
dateFormat: 'hh:mm a',
height: '450px',
dayWorkingTime: [{ from: 0, to: 23 }]
});
function actionComplete(args: any) {
    if(args.action == "TaskbarEditing") {
        console.log(args.data.ganttProperties.endDate);
    }
}
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Timezone methods

offset

This method is used to calculate the difference between the passed UTC date and timezone.

Parameters	Type	Description
----- ----- -----		
Date	Date	UTC time as date object.
Timezone	String	Timezone.

Returns **number**

```
`ts
// Assume your local timezone as IST/UTC+05:30
let timezone: Timezone = new Timezone();
let date: Date = new Date(2018,11,5,15,25,11);
let timeZoneOffset: number = timezone.offset(date,"Europe/Paris");
console.log(timeZoneOffset); //-60
`
```

convert

This method is used to convert the passed date from one timezone to another.

Parameters	Type	Description
----- ----- -----		
Date	Date	UTC time as date object.
fromOffset	number/string	Timezone from which date needs to be converted.
toOffset	number/string	Timezone to which date needs to be converted.

Returns **Date**

```
`ts
// Assume your local timezone as IST/UTC+05:30
let timezone: Timezone = new Timezone();
let date: Date = new Date(2018,11,5,15,25,11);
let convertedDate: Date = timezone.convert(date, "Europe/Paris", "Asia/Tokya");
let convertedDate1: Date = timezone.convert(date, 60, -360);
console.log(convertedDate); //2018-12-05T08:55:11.000Z
console.log(convertedDate1); //2018-12-05T16:55:11.000Z
`
```

remove

This method is used to remove the time difference between passed UTC date and timezone.

Parameters	Type	Description
----- ----- -----		
Date	Date	UTC as date object.
Timezone	String	Timezone.

Returns **Date**

`ts

// Assume your local timezone as IST/UTC+05:30

let timezone: Timezone = new Timezone();

let date: Date = new Date(2018,11,5,15,25,11);

let convertedDate: Date = timezone.remove(date, "Europe/Paris");

console.log(convertedDate); //2018-12-05T14:25:11.000Z

`

Event markers in EJ2 JavaScript Gantt control

The event markers in the Gantt control is used to highlight the important events in a project. Event markers can be initialized by using the [eventMarkers](#) property, and you can define date and label for the event markers using the [day](#) and [label](#) properties. You can also customize it using the [cssClass](#) properties. The following code example shows how to add event markers in the Gantt control.

To highlight the days, inject the [DayMarkers](#) module into the Gantt control.

INDEX.TS

```
import { Gantt, DayMarkers } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(DayMarkers);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    resourceInfo: 'resources',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  eventMarkers: [
    {
      day: '04/10/2019',
      cssClass: 'e-custom-event-marker',
      label: 'Project approval and kick-off'
    }
  ]
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt .e-gantt-chart .e-custom-event-marker {
      width: 1px;
      border-left: 2px green dotted;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Displaying eventMarkers in stacked manner

When [eventMarkers](#) are given in consecutive dates and zoomToFit is performed, they may overlap. To avoid this, you can update the position of the eventMarkers in the [dataBound](#) and [actionComplete](#) events so that they are not overlapped and are visible to read.

INDEX.TS

```

import { Gantt, DayMarkers, Toolbar } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(DayMarkers, Toolbar);
function updateEventMarker() {
  document.getElementsByClassName('e-span-label')[1].style.top = '100px';
  document.getElementsByClassName('e-gantt-right-arrow')[1].style.top =
    '110px';
}
let gantt: Gantt = new Gantt({

```

```

dataSource: GanttData,
height: '450px',
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    endDate: 'EndDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks',
},
eventMarkers: [
    {
        day: new Date('04/02/2019'),
    },
    {
        day: new Date('04/09/2019'),
        label: 'Research phase research phase research phase',
    },
    {
        day: new Date('04/10/2019'),
        label: 'Design phase',
    },
    {
        day: new Date('05/23/2019'),
        label: 'Production phase',
    },
    {
        day: new Date('06/20/2019'),
        label: 'Sales and marketing phase',
    },
],
toolbar: ['ZoomIn', 'ZoomOut', 'ZoomToFit'],
dataBound: function () {
    updateEventMarker();
    gantt.fitToProject();
},
actionComplete: function () {
    updateEventMarker();
},
projectStartDate: new Date('03/24/2019'),
projectEndDate: new Date('07/06/2019')
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<style>
.e-gantt .e-gantt-chart .e-custom-event-marker {
    width: 1px;
    border-left: 2px green dotted;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Label positions in EJ2 JavaScript gantt control

The EJ2 Gantt chart offers powerful features for customizing various labels position within the chart, enabling users to present relevant project information clearly. In EJ2 Gantt chart, `labelSettings` feature provides three key options for label customization: [rightLabel](#), [taskLabel](#), and [leftLabel](#). Label positions can be initialized by using the [labelSettings](#) property.

The following code example shows how to add label positions in the gantt control.

INDEX.TS

```

import { Gantt, Selection } from '@syncfusion/ej2-gantt';
import { GanttData, resourceCollection } from 'datasource.ts';
Gantt.Inject(Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    resources: resourceCollection,
    viewType: 'ResourceView',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        resourceInfo: 'resources',
    }
});

```

```

        work: 'work',
        child: 'subtasks'
    },
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
        unit: 'resourceUnit',
        group: 'resourceGroup'
    },
    columns: [
        { field: 'TaskID', visible: false },
        { field: 'TaskName', headerText: 'Name', width: 250 },
        { field: 'work', headerText: 'Work' },
        { field: 'Progress' },
        { field: 'resourceGroup', headerText: 'Group' },
        { field: 'StartDate' },
        { field: 'Duration' },
    ],
    labelSettings: {
        rightLabel: 'resources',
        leftLabel: 'TaskName',
        taskLabel: '${Progress}%'
    },
    splitterSettings: {
        columnIndex: 3
    },
    height: '450px',
    projectStartDate: new Date('03/22/2019'),
    projectEndDate: new Date('05/18/2019')
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>

```

```

    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Managing event marker overlapping in EJ2 JavaScript gantt control

In the EJ2 Gantt control, it is possible to customize multiple [eventMarkers](#) for the same date. However, by default, in such scenarios, these markers may overlap each other, resulting in visual clutter. To manage this, the following sample code demonstrates how to utilize the Gantt dataBound function to obtain label and arrow classes. It performs a loop action to fulfill the current requirement and to avoid overlapping. For further clarification, the code snippet below illustrates the flow of its implementation.

INDEX.TS

```

import { Gantt, DayMarkers } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(DayMarkers);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    eventMarkers: [
        {
            day: '04/10/2019',
            cssClass: 'e-custom-event-marker',
            label: 'Project approval and kick-off'
        },
        {
            day: '04/10/2019',
            cssClass: 'e-custom-event-marker',
            label: 'Project approval and kick-off'
        }
    ],
    dataBound(): void {
        let labeltop: number = 100;
        let rightarrow: number = 110;
        for (let i: number = 0; i < this.eventMarkers.length; i++) {
            (document.getElementsByClassName('e-span-label')[i] as
            HTMLElement).style.top = labeltop + 'px';
            (document.getElementsByClassName('e-gantt-right-arrow')[i] as
            HTMLElement).style.top = rightarrow + 'px';
            labeltop += 35;
            rightarrow += 35;
        }
    }
});

```

```

    }
    },
    projectStartDate: new Date('03/24/2019'),
    projectEndDate: new Date('07/06/2019')
  });
  gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt .e-gantt-chart .e-custom-event-marker {
      width: 1px;
      border-left: 2px green dotted;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Work in EJ2 JavaScript Gantt control

The work is the total hours required to complete a task. Work can be mapped from the data source field using the property [taskFields.work](#). Work can be measured in Hour, Day, Minute. By default, work is measured in Hour and it can be changed, by using the property [workUnit](#).

Note: When the work field is mapped from the data source, the default task type will be FixedWork.

INDEX.TS

```

import { Gantt, Toolbar, Edit, Selection } from '@syncfusion/ej2-gantt';
import { GanttData, resourceResources } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    resourceInfo: 'resources',
    work: 'Work',
    child: 'subtasks'
  },
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
  resources: resourceResources,
  resourceFields: {
    id: 'resourceId',
    name: 'resourceName',
    unit: 'Unit'
  },
  workUnit: 'Hour',
  toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel',
'ExpandAll', 'CollapseAll'],
  allowSelection: true,
  height: '450px',
  treeColumnIndex: 1,
  columns: [
    { field: 'TaskID', visible: false },
    { field: 'TaskName', headerText: 'Task Name', width: '180'
},
    { field: 'resources', headerText: 'Resources', width: '160'
},
    { field: 'Work', width: '110' },
    { field: 'Duration', width: '100' },
  ],
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Task type

The work, duration and resource unit fields of a task depends upon each other and will change automatically on editing any one of these fields. But we can also set these field's values as constant using the [taskType](#) property. **FixedUnit** is the default [taskType](#). The following values can be set to the [taskType](#) property,

- **FixedDuration** - Duration task field will remain constant while updating resource unit or work field.
- **FixedWork** - Work field will remain constant while updating resource unit or duration fields.
- **FixedUnit** - Resource units will remain constant while updating duration or work field.

INDEX.TS

```

import { Gantt, Toolbar, Edit, Selection } from '@syncfusion/ej2-gantt';
import { GanttData, resourceResources } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit, Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        resourceInfo: 'resources',
        work: 'Work',
        child: 'subtasks'
    },

```

```

        editSettings: {
            allowAdding: true,
            allowEditing: true,
            allowDeleting: true,
            allowTaskbarEditing: true,
            showDeleteConfirmDialog: true
        },
        resources: resourceResources,
        resourceFields: {
            id: 'resourceId',
            name: 'resourceName',
            unit: 'Unit'
        },
        workUnit: 'Hour',
        taskType: 'FixedWork',
        toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel',
'ExpandAll', 'CollapseAll'],
        allowSelection: true,
        height: '450px',
        treeColumnIndex: 1,
        columns: [
            { field: 'TaskID', visible: false },
            { field: 'TaskName', headerText: 'Task Name', width: '180'
},
            { field: 'resources', headerText: 'Resources', width: '160'
},
            { field: 'Work', width: '110' },
            { field: 'Duration', width: '100' },
            { field: 'taskType', headerText: 'Task Type', width: '110' }
        ],
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>

```

```

        </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Following table explains how the work, duration and resource unit fields will gets updated on changing any of the fields

Task Type | Changes in Duration | Changes in work | Changes in Resource Units

Fixed Duration | Work field updates | Resource unit updates | Work field updates

Fixed Work | Resource unit updates. Note: For manually scheduled task work will update. | Duration field updates. Note: For manually scheduled task resource unit updates. | Duration will update. Note: For manually scheduled task work field updates.

Fixed Unit | Work field updates | Duration field updates. Note: For manually scheduled task resource unit updates. | Duration will update. Note: For manually scheduled task work field updates.

Note: 1. Fixed Unit is the default taskType in Gantt. 2. The above calculations are not applicable for Milestones.

Resources in EJ2 JavaScript Gantt control

In Gantt, the resources are represented by staff, equipment and materials etc. In Gantt control you can show or allocate the resources (human resources) for each task.

Resource collection

The resource collection contains details about resources that are used in the project. Resources are JSON object that contains id, name, unit and group of the resources and this collection is mapped to the Gantt control using the [resources](#) property. These resource fields are mapped to the Gantt control using the [resourceFields](#) property.

Resource fields | Description

[id](#) | This field is used to assign resources to the tasks.

[name](#) | This field is used to map the resource names. These names are displayed as one of Gantt columns and also can display as labels using the [labelSettings](#) property.

[unit](#) | It indicates the amount of work that can be done by a resource for the task in a day.

[group](#) | This field is used to group the resources and the tasks assigned to that particular resource into category.

The following code snippets shows resource collection and how it assigned to Gantt control.

```
`ts
```

```
let projectResources: object[] = resources: [
```

```
{ resourceid: 1, resourceName: 'Martin Tamer', resourceGroup: 'Planning Team', resourceUnit: 50},
```

```

{ resourceid: 2, resourceName: 'Rose Fuller', resourceGroup: 'Testing Team', resourceUnit: 70 },
{ resourceid: 3, resourceName: 'Margaret Buchanan', resourceGroup: 'Approval Team' },
{ resourceid: 4, resourceName: 'Fuller King', resourceGroup: 'Development Team' },
{ resourceid: 5, resourceName: 'Davolio Fuller', resourceGroup: 'Approval Team' },
{ resourceid: 6, resourceName: 'Van Jack', resourceGroup: 'Development Team', resourceUnit: 40 },
];

let gantt: Gantt = new Gantt({
  resourceFields: {
    id: 'resourceid', //resource Id Mapping
    name: 'resourceName', //resource Name mapping
    unit: 'resourceUnit', //resource Unit mapping
    group: 'resourceGroup' //resource Group mapping
  },
  resources: projectResources //resource collection dataSource
});

gantt.appendTo('#Gantt');
`

```

Assign resource

We can assign resources for a task at initial load, using the resource id value of the resources as a collection. This collection is mapped from the dataSource to the Gantt control using the [resourceInfo](#) property.

Resources are assigned to tasks in following ways.

Assign resource alone

If the unit is not specified for specific resource, the amount of work done will be consider as 100% by default. In such cases, the resource unit will not be displayed in Gantt UI.

```

`ts
{ TaskID: 2, TaskName: 'Identify site location', StartDate: new Date('04/02/2019'), Duration: 4, Progress:
50, resources: [1] },
`

```

Assign resource with unit

We can assign the quantity of work done by the resources for the specific task as like below code snippet.

```

`ts
{ TaskID: 3, TaskName: 'Perform soil test', StartDate: new Date('03/29/2019'), Duration: 4,
resources: [{resourceid: 2, resourceUnit: 70}, {resourceid: 1, resourceUnit: 70}] },
`

```

When resource unit is defined in resource collection, the amount of work done by that particular resource will be same for all the tasks.

The following code snippet shows how to assign the resource for each task and map to Gantt control.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData, ProjectResources } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    resourceInfo: 'resources',
    child: 'subtasks'
  },
  columns: [
    { field: 'TaskID', visible: false },
    { field: 'TaskName', headerText: 'Task Name', width: '180' },
    { field: 'resources', headerText: 'Resources', width: '160' },
    { field: 'Duration', width: '100' },
  ],
  resourceFields: {
    id: 'resourceId',
    name: 'resourceName',
    unit: 'resourceUnit',
    group: 'resourceGroup'
  },
  labelSettings: {
    rightLabel: 'resources'
  },
  height: '450px',
  resources: ProjectResources,
  splitterSettings: {
    columnIndex: 5.1
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```

```

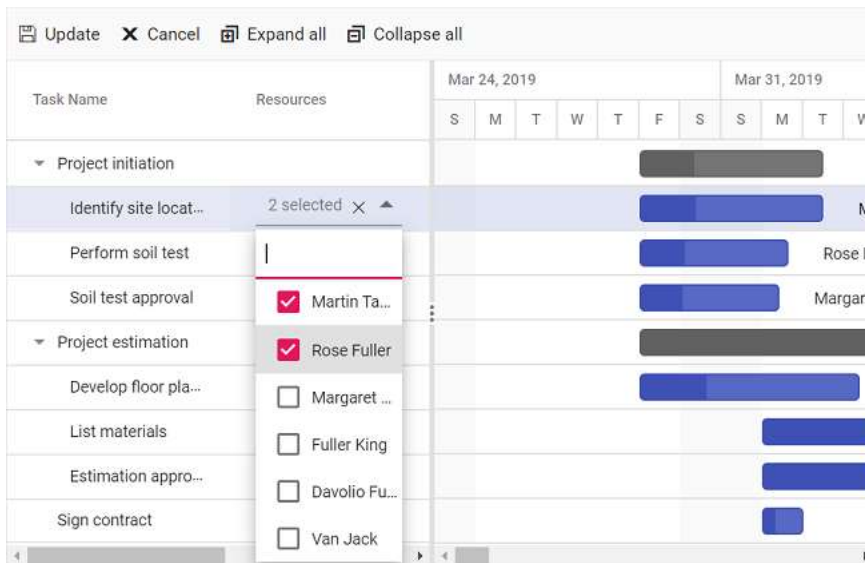
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

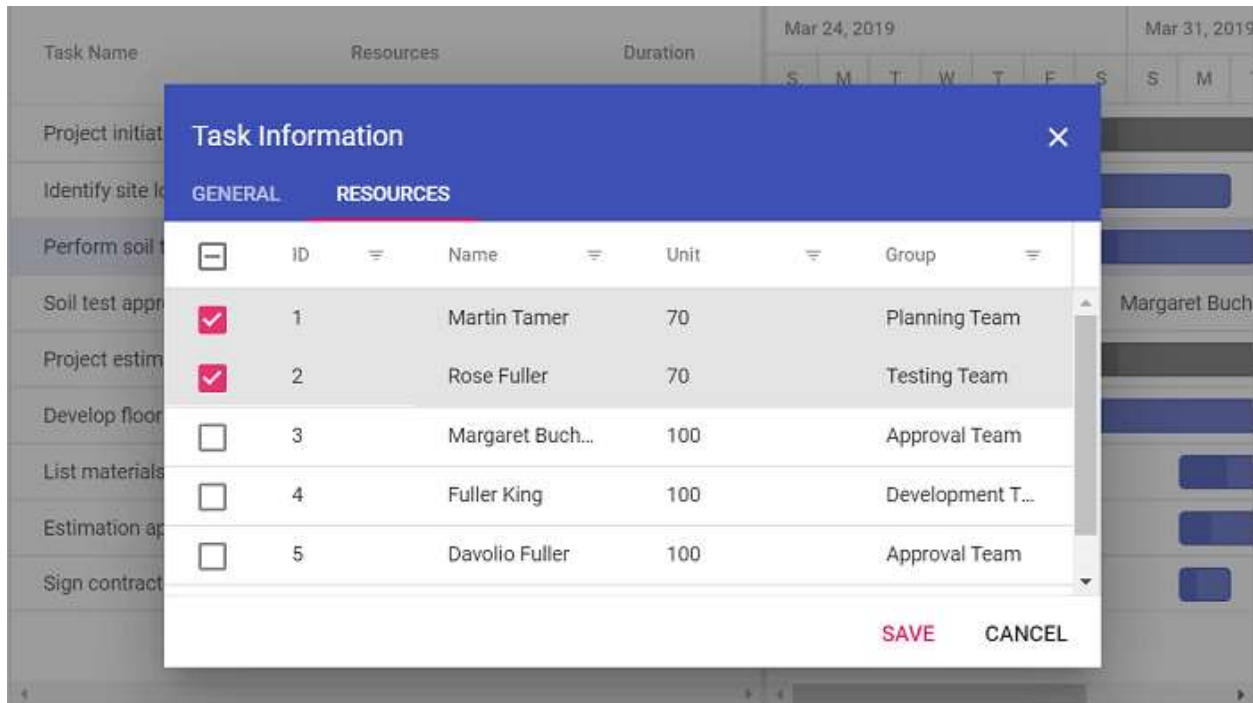
    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add/Edit resource collection

By using cell/ dialog edit option, we can add/remove the multiple resources for a particular task. Resource Unit can be change for a each task on resource tab in the edit dialog by double click on the unit cell.





Resource view in EJ2 JavaScript Gantt control

The resource breakdown view is used to visualize the tasks assigned to each resource in hierarchy manner. Resources are displayed as parents and all the tasks assigned to each resource are displayed as its child records. It can be initialized by setting the [viewType](#) property to `ResourceView`.

Unassigned task

A task not assigned to any one of the resource are termed as unassigned tasks. The unassigned tasks are grouped with a name as `Unassigned Task` and displayed at the bottom of Gantt data collection. It is validated at load time during Gantt record creation by default based on a task `resourceInfo` mapping property in the Gantt chart data source. If the resource is assigned to the unassigned grouped tasks, the task will be moved as child to the respective resource.

Resource task

A task assigned to one or more resources are termed as resource task and it is added as child task to the respective resource. Already assigned task can also be shared or moved with other resources by adding a resource name to the task or removing resource name from the task by cell or dialog editing.

Note: Currently there is no support for unscheduled task in Resource view Gantt.

INDEX.TS

```
import { Gantt, Toolbar, Edit, Selection } from '@syncfusion/ej2-gantt';
import { GanttData, resourceCollection } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  resources: resourceCollection,
  viewType: 'ResourceView',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
```

```

        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        resourceInfo: 'resources',
        work: 'work',
        child: 'subtasks'
    },
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
        unit: 'resourceUnit',
        group: 'resourceGroup'
    },
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    columns: [
        { field: 'TaskID', visible: false },
        { field: 'TaskName', headerText: 'Name', width: 250 },
        { field: 'work', headerText: 'Work' },
        { field: 'Progress' },
        { field: 'resourceGroup', headerText: 'Group' },
        { field: 'StartDate' },
        { field: 'Duration' },
    ],
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll', 'CollapseAll'],
    labelSettings: {
        rightLabel: 'resources'
    },
    splitterSettings: {
        columnIndex: 3
    },
    allowResizing: true,
    allowSelection: true,
    highlightWeekends: true,
    treeColumnIndex: 1,
    height: '450px',
    projectStartDate: new Date('03/28/2019'),
    projectEndDate: new Date('05/18/2019')
    });
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">

```



```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Resource OverAllocation

When a resource is assigned too much of work to complete within a day of resource's available time then it is called as overallocation.

The available working time of resources for completing the task in a day will be calculated based on the `dayWorkingTime` property and `resource unit`.

The range of overallocation dates can be highlighted by a square bracket. It can be enabled by setting the `showOverallocation` property as `true`. The following code example demonstrates how to hide or show the over allocation by clicking the custom button.

Note: By default, the `showOverAllocation` property value is `false`.

INDEX.TS

```

import { Gantt, Toolbar, Edit, Selection } from '@syncfusion/ej2-gantt';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { overAllocationData, resources } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit, Selection);
let gantt: Gantt = new Gantt({
    dataSource: overAllocationData,
    resources: resources,
    viewType: 'ResourceView',
    showOverAllocation: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
    }
});

```

```

        progress: 'Progress',
        dependency: 'Predecessor',
        resourceInfo: 'resources',
        work: 'work',
        child: 'subtasks'
    },
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
        unit: 'resourceUnit',
        group: 'resourceGroup'
    },
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    columns: [
        { field: 'TaskID', visible: false },
        { field: 'TaskName', headerText: 'Name', width: 250 },
        { field: 'work', headerText: 'Work' },
        { field: 'Progress' },
        { field: 'resourceGroup', headerText: 'Group' },
        { field: 'StartDate' },
        { field: 'Duration' },
    ],
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll', 'CollapseAll',
        { text: 'Show/Hide Overallocation', tooltipText: 'Show/Hide Overallocation', id: 'showhidebar' }],
    toolbarClick: (args: ClickEventArgs) => {
        if (args.item.id === 'showhidebar') {
            gantt.showOverAllocation = gantt.showOverAllocation ?
false : true;
        }
    },
    labelSettings: {
        rightLabel: 'resources',
        taskLabel: 'Progress'
    },
    projectStartDate: new Date('03/28/2019'),
    projectEndDate: new Date('05/18/2019')
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Resource Multi Taskbar

To visualize multiple tasks assigned to each resource in a row when the records are in the collapsed state. It can be enabled by settings the `enableMultiTaskbar` property value as `true`.

The collapse or expand action of a resource record can be achieved only by using the tree grid side arrow icon. Because it will be disabled on chart side action for this support.

When a resource has multiple tasks scheduled on the same date, then the tasks will be overlapped one another. Taskbar editing is also possible to change the task scheduling on the collapsed state.

Note: By default, the `enableMultiTaskbar` property value is `false`.

INDEX.TS

```

import { Gantt, Toolbar, Edit, Selection } from '@syncfusion/ej2-gantt';
import { overAllocationData, resources } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit, Selection);
let gantt: Gantt = new Gantt({
    dataSource: overAllocationData,
    resources: resources,
    viewType: 'ResourceView',
    showOverAllocation: true,
    enableMultiTaskbar: true,
    collapseAllParentTasks: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
    }
});

```

```

        dependency: 'Predecessor',
        resourceInfo: 'resources',
        work: 'work',
        expandState: 'isExpand',
        child: 'subtasks'
    },
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
        unit: 'resourceUnit',
        group: 'resourceGroup'
    },
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', headerText: 'Name', width: 250 },
        { field: 'work', headerText: 'Work' },
        { field: 'Progress' },
        { field: 'resourceGroup', headerText: 'Group' },
        { field: 'StartDate' },
        { field: 'Duration' }
    ],
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll', 'CollapseAll'],
    labelSettings: {
        rightLabel: 'resources',
        taskLabel: 'TaskName'
    },
    projectStartDate: new Date('03/28/2019'),
    projectEndDate: new Date('05/18/2019')
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable taskbar drag and drop

In Gantt, you can enable taskbar drag and drop between resources by using the [allowTaskbarDragAndDrop](#) property. This allows you to move a taskbar from one resource to another vertically, making it easier to schedule tasks and manage resources.

INDEX.TS

```

import { Gantt, Toolbar, Edit, Selection } from '@syncfusion/ej2-gantt';
import { overAllocationData, resources } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit, Selection);
let gantt: Gantt = new Gantt({
    dataSource: overAllocationData,
    resources: resources,
    viewType: 'ResourceView',
    showOverAllocation: true,
    enableMultiTaskbar: true,
    allowTaskbarDragAndDrop: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        resourceInfo: 'resources',
        work: 'work',
        expandState: 'isExpand',
        child: 'subtasks'
    },
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
        unit: 'resourceUnit',
        group: 'resourceGroup'
    },
    editSettings: {
        allowAdding: true,

```

```

        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', headerText: 'Name', width: 250 },
        { field: 'work', headerText: 'Work' },
        { field: 'Progress' },
        { field: 'resourceGroup', headerText: 'Group' },
        { field: 'StartDate' },
        { field: 'Duration' },
    ],
    toolbar: ['ExpandAll', 'CollapseAll'],
    labelSettings: {
        rightLabel: 'resources',
        taskLabel: 'TaskName'
    },
    projectStartDate: new Date('03/28/2019'),
    projectEndDate: new Date('05/18/2019')
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Disable taskbar overlap

In Gantt, you can disable taskbar overlap between resource tasks using the [allowTaskbarOverlap](#) property. This prevents the taskbars for different tasks from overlapping on the same row, making it easier to distinguish between the different tasks and manage resources effectively.

When `allowTaskbarOverlap` is set to false, the resources are displayed in a single row and the row height will be extended to occupy the tasks of the resource when it is in a collapsed state. This view allows you to easily identify any overallocation of tasks for a resource in a project.

It's important to note that when `allowTaskbarOverlap` is disabled, task dependencies or relationships cannot be established between tasks that are rendered in multiple lines for the same resource. If you need to establish dependencies between tasks for the same resource, you may want to consider enabling taskbar overlap.

INDEX.TS

```
import { Gantt, Toolbar, Edit, Selection } from '@syncfusion/ej2-gantt';
import { overAllocationData, resources } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: overAllocationData,
  resources: resources,
  viewType: 'ResourceView',
  showOverAllocation: true,
  enableMultiTaskbar: true,
  allowTaskbarOverlap: false,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    endDate: 'EndDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    resourceInfo: 'resources',
    work: 'work',
    expandState: 'isExpand',
    child: 'subtasks'
  },
  resourceFields: {
    id: 'resourceId',
    name: 'resourceName',
    unit: 'resourceUnit',
    group: 'resourceGroup'
  },
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
  columns: [
```

```

        { field: 'TaskID' },
        { field: 'TaskName', headerText: 'Name', width: 250 },
        { field: 'work', headerText: 'Work' },
        { field: 'Progress' },
        { field: 'resourceGroup', headerText: 'Group' },
        { field: 'StartDate' },
        { field: 'Duration' },
    ],
    toolbar: ['ExpandAll', 'CollapseAll'],
    labelSettings: {
        rightLabel: 'resources',
        taskLabel: 'TaskName'
    },
    projectStartDate: new Date('03/28/2019'),
    projectEndDate: new Date('05/18/2019')
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Holidays in EJ2 JavaScript Gantt control

Non-working days in a project can be displayed in the Gantt control using the [holidays](#) property. Each holiday can be defined with the following properties:

- [from](#): Defines start date of the holiday(s).
- [to](#): Defines end date of the holiday(s).
- [label](#): Defines the description or label for the holiday.
- [cssClass](#): Formats the holidays label in the Gantt chart.

To highlight the holidays, inject the [DayMarkers](#) module into the Gantt control.

The following code example shows how to display the non-working days in the Gantt control.

INDEX.TS

```
import { Gantt, DayMarkers } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(DayMarkers);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  holidays: [{
    from: "04/04/2019",
    to: "04/05/2019",
    label: " Public holidays",
    cssClass: "e-custom-holiday"
  },
  {
    from: "04/12/2019",
    to: "04/12/2019",
    label: " Public holiday",
    cssClass: "e-custom-holiday"
  }
  ]
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt .e-gantt-chart .e-custom-holiday {
```

```

        background-color:#e82869;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip in EJ2 JavaScript Gantt control

The Gantt control has a support to display a tooltip for various UI elements like taskbar, timeline cells, and grid cells

Enable tooltip

In the Gantt control, you can enable or disable the mouse hover tooltip for the following UI elements using the [tooltipSettings.showTooltip](#) property:

- Taskbar
- Connector line
- Baseline
- Event marker

INDEX.TS

```

import { Gantt, DayMarkers } from '@syncfusion/ej2-gantt';
Gantt.Inject(DayMarkers);
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site
location', BaselineStartDate: new Date('04/02/2019'), BaselineEndDate: new
Date('04/02/2019'), StartDate: new Date('04/02/2019'), Duration: 0,
Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), BaselineStartDate: new

```

```

Date('04/04/2019'),BaselineEndDate: new Date('04/09/2019'), Duration: 4,
Progress: 50 },
    { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'),BaselineStartDate: new
Date('04/08/2019'),BaselineEndDate: new Date('04/12/2019'), Duration:
4,Predecessor:"2FS", Progress: 50 },
    ]
},
{
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
        { TaskID: 6, TaskName: 'Develop floor plan for
estimation',BaselineStartDate: new Date('04/04/2019'),BaselineEndDate: new
Date('04/08/2019'), StartDate: new Date('04/04/2019'), Duration: 3,
Progress: 50 },
        { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'),BaselineStartDate: new
Date('04/02/2019'),BaselineEndDate: new Date('04/04/2019'), Duration: 3,
Progress: 50 },
        { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/02/2019'),BaselineStartDate: new
Date('04/02/2019'),BaselineEndDate: new Date('04/08/2019'), Duration:
0,Predecessor:"6SS", Progress: 50 }
    ]
},
];
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        baselineStartDate: "BaselineStartDate",
        baselineEndDate: "BaselineEndDate",
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    eventMarkers: [
        {
            day: '04/10/2019',
            label: 'Project approval and kick-off'
        }
    ],
    renderBaseline: true,
    baselineColor: 'red',
    tooltipSettings: {
        showTooltip: true
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

The default value of the [tooltipSettings.showTooltip](#) property is true.

Timeline cells tooltip

In the Gantt control, you can enable or disable the mouse hover tooltip of timeline cells using the [timelineSettings.showTooltip](#) property. The default value of this property is true. The following code example shows how to enable the timeline cells tooltip in Gantt.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',

```

```

        child: 'subtasks'
      },
      timelineSettings: {
        showTooltip: true
      }
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell tooltip

You can enable or disable the Grid cell tooltip using the [columns.clipMode](#) property.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',

```

```

        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
        { field: 'TaskName', headerText: 'Task Name', width: '150',
clipMode: 'EllipsisWithTooltip' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150', clipMode:
'Clip' },
        { field: 'Progress', headerText: 'Progress', width: '150' }
    ]
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Clip mode

The clip mode provides options to display its overflow cell content and it can be defined by the [columns.clipMode](#) property.

The following are three types of `clipMode`:

- **Clip**: Truncates the cell content when it overflows its area.
- **Ellipsis**: Displays ellipsis when content of the cell overflows its area.
- **EllipsisWithTooltip**: Displays ellipsis when content of the cell overflows its area; it displays the tooltip content when hover over ellipsis.

NOTE

By default, all the column's `clipMode` property is defined as `EllipsisWithTooltip`.

Tooltip template

Taskbar tooltip

The default tooltip in the Gantt control can be customized using the `tooltipSettings.taskbar` property. You can map the template script element's ID value or template string directly to this property.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  tooltipSettings: {
    showTooltip: true,
    taskbar: '#taskbarTooltip'
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script type="text/x-jsrender" id="taskbarTooltip">
    <div>TaskID: ${TaskID}</div>
  </script>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Connector line tooltip

The default connector line tooltip in the Gantt control can be customized using the [tooltipSettings.connectorLine](#) property. You can map the value to this property as template script element ID or template string format. The following code example shows how to use the [tooltipSettings.connectorLine](#) property.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  tooltipSettings: {
    showTooltip: true,
    connectorLine: '#dependencyLineTooltip'
  }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```



```

<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <script type="text/x-jsrender" id="dependencyLineTooltip">
    <div>Offset : ${offsetString}</div>
  </script>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Taskbar editing tooltip

The taskbar editing tooltip can be customized using the [tooltipSettings.editing](#) property. The following code example shows how to customize the taskbar editing tooltip in Gantt.

INDEX.TS

```

import { Gantt, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  editSettings: {
    allowEditing: true,
    allowTaskbarEditing: true
  },
  tooltipSettings: {

```

```

        showTooltip: true,
        editing: '#editingTooltip'
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script type="text/x-jsrender" id="editingTooltip">
    <div>Duration : ${duration}</div>
  </script>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Baseline tooltip

A baseline tooltip can be customized using the [tooltipSettings.baseline](#) property. The following code example shows how to customize the baseline tooltip in Gantt.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),

```

```

        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site
location',BaselineStartDate: new Date('04/02/2019'),BaselineEndDate: new
Date('04/02/2019'), StartDate: new Date('04/02/2019'), Duration: 0,
Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'),BaselineStartDate: new
Date('04/04/2019'),BaselineEndDate: new Date('04/09/2019'), Duration: 4,
Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'),BaselineStartDate: new
Date('04/08/2019'),BaselineEndDate: new Date('04/12/2019'), Duration: 4,
Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for
estimation',BaselineStartDate: new Date('04/04/2019'),BaselineEndDate: new
Date('04/08/2019'), StartDate: new Date('04/04/2019'), Duration: 3,
Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'),BaselineStartDate: new
Date('04/02/2019'),BaselineEndDate: new Date('04/04/2019'), Duration: 3,
Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/02/2019'),BaselineStartDate: new
Date('04/02/2019'),BaselineEndDate: new Date('04/08/2019'), Duration: 0,
Progress: 50 }
        ]
    },
];
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        baselineStartDate: "BaselineStartDate",
        baselineEndDate: "BaselineEndDate",
        progress: 'Progress',
        child: 'subtasks'
    },
    tooltipSettings: {
        showTooltip: true,
        baseline: '#baselineTooltip'
    },
    renderBaseline: true,
    baselineColor: 'red'
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <script type="text/x-jsrender" id="baselineTooltip">
    <div>Baseline StartDate :
    ${this.getFormattedDate(BaselineStartDate)}</div>
  </script>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Appearance customization in EJ2 JavaScript Gantt control

Taskbar customization

Taskbar Height

Height of child taskbars and parent taskbars can be customized by using [taskbarHeight](#) property. The following code example shows how to use the property.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',

```

```

        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    taskbarHeight: 50,
    rowHeight: 60
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

NOTE

The [taskbarHeight](#) value should be lower than the [rowHeight](#) property value and these properties accept only pixel values.

Conditional formatting

The default taskbar UI can be replaced with custom templates using the [queryTaskbarInfo](#) event. The following code example shows customizing the taskbar UI based on task progress values in the Gantt control.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 70 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 80 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 70 }
        ]
    },
];
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    queryTaskbarInfo: function (args: any) {
        if (args.data.Progress == 50) {
            args.progressBarBgColor = "red";
        } else if (args.data.Progress == 70) {
            args.progressBarBgColor = "yellow";
        } else if (args.data.Progress == 80) {
            args.progressBarBgColor = "lightgreen";
        }
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Taskbar template

You can design your own taskbars to view the tasks in Gantt by using [taskbarTemplate](#) property. And it is possible to map the template script element's ID value to this property. It is also possible to customize the parent taskbars and milestones with custom templates by using [parentTaskbarTemplate](#) and [milestoneTemplate](#) properties.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  rowHeight: 60,

```

```

milestoneTemplate: '#MilestoneTemplate',
parentTaskbarTemplate: '#ParentTaskbarTemplate',
taskbarTemplate: '#TaskbarTemplate'
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script type="text/x-jsrender" id="TaskbarTemplate">
    <div class="e-gantt-child-taskbar-inner-div e-gantt-child-taskbar"
style="height:100%">
      <div class="e-gantt-child-progressbar-inner-div e-gantt-child-
progressbar" style="width:${ganttProperties.progressWidth}px;height:100%">
        <span class="e-task-label" style="position: absolute; z-index: 1;
font-size: 12px; color: white; top: 5px; left: 10px; font-family: "Segoe
UI"; overflow: hidden; text-overflow: ellipsis; width: 40%; cursor:
move;">${taskData.TaskName}</span>
      </div>
    </div>
  </script>
<script type="text/x-jsrender" id="ParentTaskbarTemplate">
  <div class="e-gantt-parent-taskbar-inner-div e-gantt-parent-taskbar"
style="height:100%">
    <div class="e-gantt-parent-progressbar-inner-div e-gantt-parent-
progressbar" style="width:${ganttProperties.progressWidth}px;height:100%">
      <span class="e-task-label" style="position: absolute; z-index: 1;
font-size: 12px; color: white; top: 5px; left: 10px; font-family: "Segoe
UI"; overflow: hidden; text-overflow: ellipsis; width: 40%; cursor:
move;">${taskData.TaskName}</span>
    </div>
  </div>
</script>
<script type="text/x-jsrender" id="MilestoneTemplate">
  <div class="e-gantt-milestone" style="position:absolute;">
    <div class="e-milestone-top" style="border-right-width:15px;border-
left-width:15px;border-bottom-width:15px;"></div>
    <div class="e-milestone-bottom" style="top:15px;border-right-
width:15px; border-left-width:15px; border-top-width:15px;">

```



```

        </div>
    </div>
</script>

    <div id="container">
        <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change gripper icon in taskbar

You can change the gripper icon in the taskbar by applying styles to their respective class elements.

INDEX.TS

```

import { Gantt, Edit, Filter, Sort } from '@syncfusion/ej2-gantt';
Gantt.Inject(Edit, Filter, Sort);
let projectResources: Object[] = [
    { resourceId: 1, resourceName: 'Martin Tamer' },
    { resourceId: 2, resourceName: 'Rose Fuller' },
    { resourceId: 3, resourceName: 'Margaret Buchanan' },
    { resourceId: 4, resourceName: 'Fuller King' },
    { resourceId: 5, resourceName: 'Davolio Fuller' },
    { resourceId: 6, resourceName: 'Van Jack' },
    { resourceId: 7, resourceName: 'Fuller Buchanan' },
    { resourceId: 8, resourceName: 'Jack Davolio' },
    { resourceId: 9, resourceName: 'Tamer Vinet' },
    { resourceId: 10, resourceName: 'Vinet Fuller' },
    { resourceId: 11, resourceName: 'Bergs Anton' },
    { resourceId: 12, resourceName: 'Construction Supervisor' }
];
let ganttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, resources: [2, 3, 5] },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Predecessor: "2FS", Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),

```

```

        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50, resources: [4]
},
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, resources: [4, 8], },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0,Predecessor:"6SS", Progress: 50, resources:
[12, 5] }
        ]
    },
    ],
let gantt: Gantt = new Gantt({
    dataSource: ganttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        resourceInfo: 'resources',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width: '250' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'resources', headerText: 'Resources', width: '200' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
        { field: 'Predecessor', headerText: 'Predecessor', width: '150'
    },
    ],
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
    },
    resources: projectResources,
    allowSorting: true,
    allowFiltering: true,
    editSettings: {
        allowEditing: true,
        allowTaskbarEditing: true
    }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Task labels

The Gantt control maps any data source fields to task labels using the [labelSettings.leftLabel](#), [labelSettings.rightLabel](#), and [labelSettings.taskLabel](#) properties. You can customize the task labels with templates.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    labelSettings: {
        leftLabel: 'TaskID',
        rightLabel: 'Task Name: ${taskData.TaskName}',
        taskLabel: '${Progress}%'
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Connector lines

The width and background color of connector lines in Gantt can be customized using the [connectorLineWidth](#) and [connectorLineBackground](#) properties. The following code example shows how to use these properties.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
      { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
      { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Predecessor:"3SS", Duration: 4, Progress: 50 },
    ]
  }
]

```

```

    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
              StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
              Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
              Date('04/04/2019'), Predecessor: "7FS", Duration: 3, Progress: 50 }
        ]
    },
    ];
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        dependency: 'Predecessor',
        progress: 'Progress',
        child: 'subtasks'
    },
    connectorLineBackground: "red",
    connectorLineWidth: 3
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
    rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

    <div id="Gantt"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize rows and cells

While rendering the TreeGrid part in Gantt, the [rowDataBound](#) and [queryCellInfo](#) events trigger for every row and cell. Using these events, you can customize the rows and cells. The following code example shows how to customize the cell and row elements using these events.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 0, Progress: 80 },
      { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
      { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
    ]
  },
  {
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
      { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 60 },
      { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0, Progress: 50 }
    ]
  },
];
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',

```

```

        duration: 'Duration',
        dependency: 'Predecessor',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
    ],
    splitterSettings: {
        columnIndex: 3
    },
    queryCellInfo: function (args: any) {
        if (args.column.field == "Progress") {
            if (args.data.Progress < 60)
                args.cell.style.backgroundColor = "lightgreen"
            else
                args.cell.style.backgroundColor = "yellow"
        }
    },
    rowDataBound: function (args: any) {
        if (args.data.TaskID == 4)
            args.row.style.backgroundColor = "red"
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Grid lines

In the Gantt control, you can show or hide the grid lines in the TreeGrid side and chart side by using the [gridLines](#) property.

The following options are available in the Gantt control for rendering the grid lines:

- Horizontal: The horizontal grid lines alone will be visible.
- Vertical: The vertical grid lines alone will be visible.
- Both: Both the horizontal and vertical grid lines will be visible on the TreeGrid and chart sides.
- None: Gridlines will not be visible on TreeGrid and chart sides.

By default, the [gridLines](#) property is set to **Horizontal** type.

The following code example shows how to change the gridlines rendering mode in the Gantt control.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    gridLines: 'Both'
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Gantt</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Splitter

In the Gantt control, the Splitter separates the TreeGrid section from the Chart section. You can change the position of the Splitter when loading the Gantt control using the [splitterSettings](#) property. By splitting the TreeGrid from the chart, the width of the TreeGrid and chart sections will vary in the control. The [splitterSettings.position](#) property denotes the percentage of the TreeGrid section's width to be rendered and this property supports both pixels and percentage values. You can define the splitter position as column index value using the [splitterSettings.columnIndex](#) property. You can also define the splitter position with built-in splitter view modes by using the [splitterSettings.view](#) property. The following list is the possible values for this property:

- **Default:** Shows Grid side and Gantt side.
- **Grid:** Shows Grid side alone in Gantt.
- **Chart:** Shows chart side alone in Gantt.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { data } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: data,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings: {

```

```

        position: "50%"
    }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Change splitter position dynamically

In Gantt, we can change the splitter position dynamically by using [setSplitterPosition](#) method. We can change the splitter position by passing value and type parameter to [setSplitterPosition](#) method. Type parameter will accept one of the following values 'position', 'columnIndex', 'viewType'. The following code example shows how to use this method.

INDEX.TS

```

import { Gantt, Edit, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
import { data } from 'datasource.ts';
Gantt.Inject(Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: data,
  height: '450px',
  taskFields: {
```

```

        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    }
});
gantt.appendTo('#Gantt');
let splitBtn: Button = new Button();
splitBtn.appendTo('#changeByPosition');
let splittBtn: Button = new Button();
splittBtn.appendTo('#changeByIndex');
document.getElementById('changeByPosition').addEventListener('click', () => {
    gantt.setSplitterPosition('50%', 'position');
});
document.getElementById('changeByIndex').addEventListener('click', () => {
    gantt.setSplitterPosition(0, 'columnIndex');
});
let dropDownMode: DropDownList = new DropDownList({
    dataSource: [
        { id: 'Default', mode: 'Default' },
        { id: 'Grid', mode: 'Grid' },
        { id: 'Chart', mode: 'Chart' },
    ],
    fields: { text: 'mode', value: 'id' },
    value: 'Default',
    change: (e: ChangeEventArgs) => {
        let viewType: any = <string>e.value;
        gantt.setSplitterPosition(viewType, 'view');
    }
});
dropDownMode.appendTo('#view');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
    <div>
        <div style="padding-top: 7px; display: inline-block">Change Splitter
View</div>
        <div style="display: inline-block">
            <input type="text" id="view">
        </div>
    </div>
    <button id="changeByPosition">Change Splitter By
Position</button> <button id="changeByIndex">Change Splitter By
ColumnIndex</button>
    <div id="Gantt"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Duration unit in EJ2 JavaScript Gantt control

Duration units

In Gantt, the task's duration value can be measured by the following duration units,

- Day
- Hour
- Minute

In Gantt, we can define duration unit for whole project by using [durationUnit](#) property, when we defines the value for this property, this unit will be applied for all task which don't has duration unit value. And each task in the project can be defined with different duration units and the duration unit of a task can be defined by the following ways,

- Using [taskFields.durationUnit](#) property, to map the duration unit data source field.
- Defining the duration unit value along with the duration field in the data source.

Mapping the duration unit field

The below code snippet explains the mapping of duration unit data source field to the Gantt control using the [taskFields.durationUnit](#) property.

INDEX.TS

```

let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [

```

```

        { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 2, DurationUnit:'day', Progress: 50 },
        { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 12,DurationUnit:'hour', Progress: 70 },
        { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 240,DurationUnit:'minute', Progress: 80 },
    ]
},
{
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
        { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, DurationUnit:'hour',
Progress: 50 },
        { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, DurationUnit:'day' },
        { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 480, DurationUnit:'minute', Progress: 70 }
    ]
},
];
import { Gantt } from '@syncfusion/ej2-gantt';
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        durationUnit: 'DurationUnit',
        child: 'subtasks'
    },
    splitterSettings: {
        columnIndex: 4
    }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

NOTE

The default value of the [durationUnit](#) property is `day`.

Defining duration unit along with duration field

Duration units for the tasks can also be defined along with the duration values, the below code snippet explains the duration unit for a task along with duration value,

INDEX.TS

```

let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent: true,
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: '3days', Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: '12hours', Progress: 70 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: '1800minutes', Progress: 80 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        isParent: true,
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: '12hours', Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: '3days', Progress: 50 },
        ]
    }
]

```

```

        { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: '480minutes', Progress: 70 }
    ]
    },
    ];
import { Gantt } from '@syncfusion/ej2-gantt';
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings: {
        columnIndex: 4
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

NOTE:

The edit type of the duration column in Gantt is string, to support editing the duration field along with duration units.

Task scheduling in EJ2 JavaScript Gantt control

The Gantt provides support for automatic and manual task scheduling modes. It is used to indicate whether the start date and end date of all the tasks will be automatically validated or not. [taskMode](#) is the property used to change the schedule mode of a task.

The Gantt control supports three types of mode. They are:

- **Auto:** All the tasks are automatically validate.
- **Manual:** All the tasks are manually validate by the user.
- **Custom:** Both Auto and Manual tasks are render by mapped from data source.

Note: The default value of [taskMode](#) is **Auto**.

Automatically Scheduled Tasks

When the [taskMode](#) property is set as **Auto**, the start date and end date of all the tasks in the project will be automatically validated. That is, dates are validated based on various factors such as working time, holidays, weekends and predecessors.

INDEX.TS

```
import { Gantt, Toolbar, Edit, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    endDate: 'EndDate',
    dependency: 'Predecessor',
    child: 'Children',
  },
  height: '450px',
  taskMode: 'Auto',
  toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
    'CollapseAll', 'Search'],
  treeColumnIndex: 1,
  editSettings: {
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
});
gantt.appendTo('#Gantt');
```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Manually Scheduled Tasks

When the [taskMode](#) property is set as **Manual**, the start date and end date of all the tasks in the project will be same as given in the data source. That is, dates are not validated based on various factors such as dependencies between tasks, holidays, weekends, working time. We can restrict this mode in predecessor validation alone. That is, we can automatically validate the dates based on predecessor values by enabling the [validateManualTasksOnLinking](#) property.

INDEX.TS

```

import { Gantt, Toolbar, Edit, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    endDate: 'EndDate',
    dependency: 'Predecessor',

```

```

        child: 'Children',
    },
    height: '450px',
    taskMode : 'Manual',
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
    'CollapseAll', 'Search'],
    validateManualTasksOnLinking: true,
    treeColumnIndex: 1,
    editSettings: {
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    });
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom

When the [taskMode](#) property is set as **Custom**, the scheduling mode for each tasks will be mapped from the data source field. The **Boolean** property [taskFields.manual](#) is used to map the manual scheduling mode field from the data source.

INDEX.TS

```
import { Gantt, Toolbar, Edit, Selection } from '@syncfusion/ej2-gantt';
Gantt.Inject(Toolbar, Edit, Selection);
let GanttData: Object[] = [
    {
        'TaskID': 1,
        'TaskName': 'Parent Task 1',
        'StartDate': new Date('02/27/2017'),
        'EndDate': new Date('03/03/2017'),
        'Progress': '40',
        'isManual': true,
        'Children': [
            { 'TaskID': 2, 'TaskName': 'Child Task 1', 'StartDate': new
Date('02/27/2017'),
                'EndDate': new Date('03/03/2017'), 'Progress': '40' },
            { 'TaskID': 3, 'TaskName': 'Child Task 2', 'StartDate': new
Date('02/26/2017'),
                'EndDate': new Date('03/03/2017'), 'Progress': '40',
                'isManual': true },
            { 'TaskID': 4, 'TaskName': 'Child Task 3', 'StartDate': new
Date('02/27/2017'),
                'EndDate': new Date('03/03/2017'), 'Duration': 5, 'Progress':
'40', }
        ]
    },
    {
        'TaskID': 5,
        'TaskName': 'Parent Task 2',
        'StartDate': new Date('03/05/2017'),
        'EndDate': new Date('03/09/2017'),
        'Progress': '40',
        'isManual': true,
        'Children': [
            { 'TaskID': 6, 'TaskName': 'Child Task 1', 'StartDate': new
Date('03/06/2017'),
                'EndDate': new Date('03/09/2017'), 'Progress': '40' },
            { 'TaskID': 7, 'TaskName': 'Child Task 2', 'StartDate': new
Date('03/06/2017'),
                'EndDate': new Date('03/09/2017'), 'Progress': '40', },
            { 'TaskID': 8, 'TaskName': 'Child Task 3', 'StartDate': new
Date('02/28/2017'),
                'EndDate': new Date('03/05/2017'), 'Progress': '40',
                'isManual': true },
            { 'TaskID': 9, 'TaskName': 'Child Task 4', 'StartDate': new
Date('03/04/2017'),
                'EndDate': new Date('03/09/2017'), 'Progress': '40',
                'isManual': true }
        ]
    },
    {

```

```

        'TaskID': 10,
        'TaskName': 'Parent Task 3',
        'StartDate': new Date('03/13/2017'),
        'EndDate': new Date('03/17/2017'),
        'Progress': '40',
        'Children': [
            { 'TaskID': 11, 'TaskName': 'Child Task 1', 'StartDate': new
Date('03/13/2017'),
              'EndDate': new Date('03/17/2017'), 'Progress': '40' },
            { 'TaskID': 12, 'TaskName': 'Child Task 2', 'StartDate': new
Date('03/13/2017'),
              'EndDate': new Date('03/17/2017'), 'Progress': '40' },
            { 'TaskID': 13, 'TaskName': 'Child Task 3', 'StartDate': new
Date('03/13/2017'),
              'EndDate': new Date('03/17/2017'), 'Progress': '40' },
            { 'TaskID': 14, 'TaskName': 'Child Task 4', 'StartDate': new
Date('03/12/2017'),
              'EndDate': new Date('03/17/2017'), 'Progress': '40' },
            { 'TaskID': 15, 'TaskName': 'Child Task 5', 'StartDate': new
Date('03/13/2017'),
              'EndDate': new Date('03/17/2017'), 'Progress': '40' }
        ]
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        endDate: 'EndDate',
        dependency: 'Predecessor',
        child: 'Children',
        manual: 'isManual',
    },
    taskMode : 'Custom',
    height: '450px',
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
'CollapseAll', 'Search'],
    columns: [
        { field: 'TaskID', visible: false },
        { field: 'TaskName' },
        { field: 'isManual' }
    ],
    validateManualTasksOnLinking: true,
    treeColumnIndex: 1,
    editSettings: {
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>

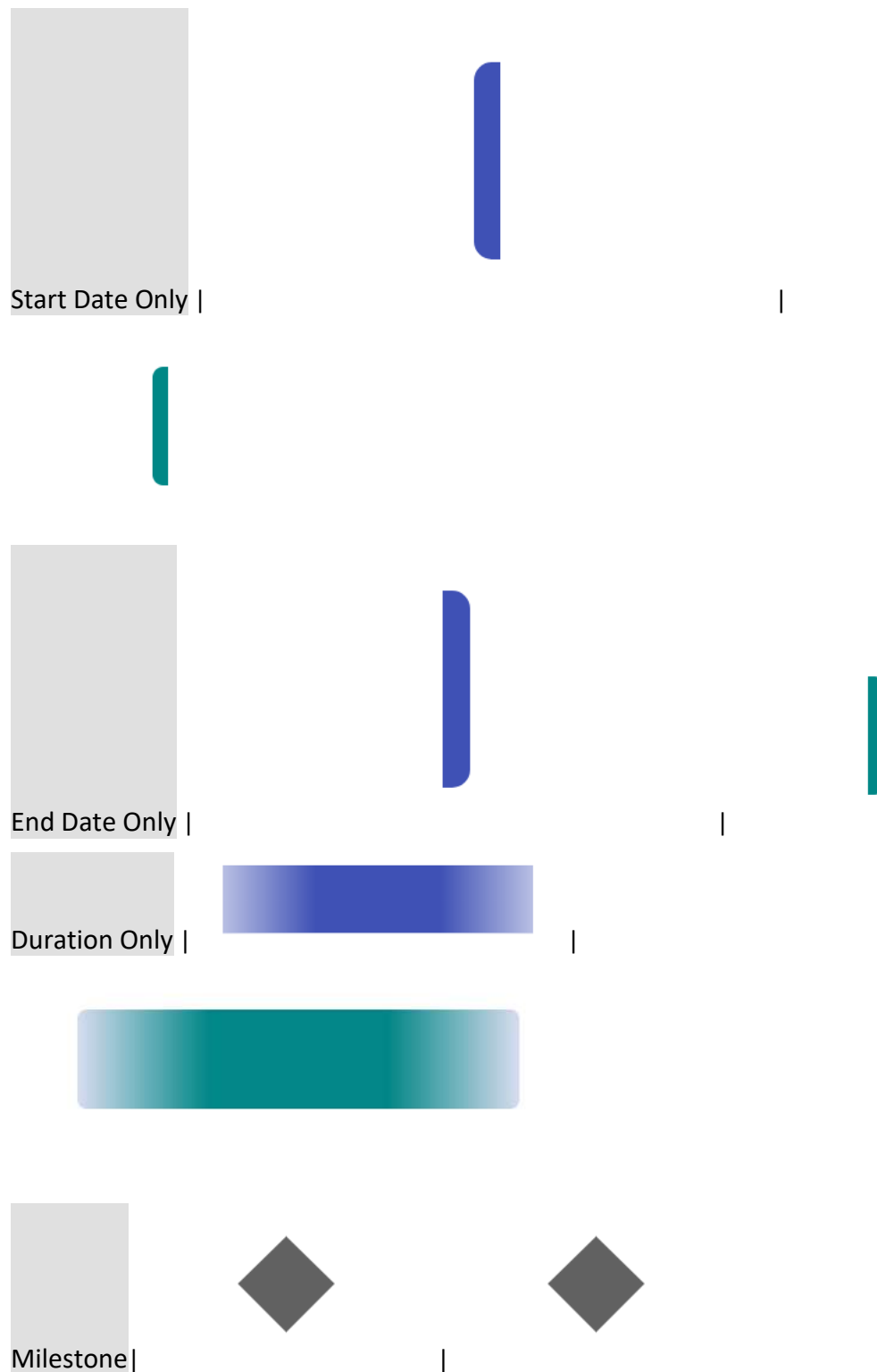
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Unscheduled Tasks

Unscheduled tasks are planned for a project without any definite schedule dates. The Gantt control supports rendering the unscheduled tasks. You can create or update the tasks with anyone of start date, end date, and duration values or none. You can enable or disable the unscheduled tasks by using the [allowUnscheduledTasks](#) property. The following images represent the various types of unscheduled tasks in Gantt.

Taskbar state | Auto | Manual



Note: A milestone is a task that has no start and end dates, but it has a duration value of zero

[Define unscheduled tasks in data source](#)

You can define the various types of unscheduled tasks in the data source as follows

INDEX.TS

```

import { Gantt, Edit } from '@syncfusion/ej2-gantt';
Gantt.Inject(Edit);
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', Duration: 3,
Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', EndDate: new
Date('04/08/2019'), Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), EndDate: new Date('04/08/2019'),
Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', Duration: 0,
Progress: 50 }
        ]
    },
];
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowEditing: true,
        allowTaskbarEditing: true
    },
    allowUnscheduledTasks: true
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Gantt</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

NOTE

If the [allowUnscheduledTasks](#) property is set to false, then the Gantt control automatically calculates the scheduled date values with a default value of duration 1 and the project start date is considered as the start date for the task.

Working Time Range

In the Gantt control, working hours in a day for a project can be defined by using the [dayWorkingTime](#) property. Based on the working hours, automatic date scheduling and duration validations for a task are performed.

The following code snippet explains how to define the working time range for the project in Gantt.

INDEX.TS

```

import { Gantt, DayMarkers } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(DayMarkers);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',

```



```

        progress: 'Progress',
        child: 'subtasks'
    },
    highlightWeekends: true,
    dayWorkingTime: [{ from: 9, to: 18 }],
    timelineSettings: {
        timelineViewMode: 'Day'
    },
    splitterSettings: {
        columnIndex: 0
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

NOTE

- * Individual tasks can lie between any time within the defined working time range of the project.
- * The [dayWorkingTime](#) property is used to define the working time for the whole project.

Weekend/Non-working days

Non-working days/weekend are used to represent the non-productive days in a project. You can define the non-working days in a week using the [workWeek](#) property in Gantt.

INDEX.TS

```
import { Gantt, DayMarkers } from '@syncfusion/ej2-gantt';
import { data } from 'datasource.ts';
Gantt.Inject(DayMarkers)
let gantt: Gantt = new Gantt({
  dataSource: data,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  workWeek: ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday"],
  highlightWeekends: true
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

By default, Saturdays and Sundays are considered as non-working days/weekend in a project.

In the Gantt control, you can make weekend as working day by setting the [includeWeekend](#) property to `true`.

Critical path in EJ2 JavaScript Gantt control

The critical path in a project is indicated by a single task or a series of tasks. If a task in critical path is delayed, the entire project will be delayed. A task is considered to be critical if any delay to this task would affect the project end date.

The critical path can be enabled in Gantt by using the built-in toolbar button or [enableCriticalPath](#) property.

The following code example shows how to display the critical path in the Gantt control:

INDEX.TS

```
import { Gantt, CriticalPath, Toolbar, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(CriticalPath, Toolbar, Edit);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  enableCriticalPath: true,
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
  toolbar: ['CriticalPath'],
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
```

```

<style>
.e-gantt .e-gantt-chart .e-custom-holiday {
    background-color:#e82869;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize taskbar in critical path

The taskbar in critical path can be customized by using [queryTaskbarInfo](#) event and [isCritical](#) property of row [data](#) in the event argument.

The following code example shows how to customize the critical path taskbar in the Gantt control:

INDEX.TS

```

import { Gantt, CriticalPath, Toolbar, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(CriticalPath, Toolbar, Edit);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    enableCriticalPath: true,
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
    }
});

```

```

        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    toolbar: ['CriticalPath'],
    queryTaskbarInfo(args: any) {
        if (args.data.isCritical && !args.data.hasChildRecords) {
            args.taskbarBgColor = 'rgb(242, 210, 189)';
            args.progressBarBgColor = 'rgb(201, 169, 166)';
        }
    }
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <style>
        .e-gantt .e-gantt-chart .e-custom-holiday {
            background-color:#e82869;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Rows in EJ2 JavaScript Gantt control

Row represents a task information from the data source, and it is possible to perform the following actions in Gantt rows.

Row height

It is possible to change the height of the row in Gantt by setting row height in pixels to the [rowHeight](#) property. The following code example explains how to change the row height in Gantt at load time.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  rowHeight: 60
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Expand/Collapse Row

In Gantt parent tasks are expanded/collapsed by using expand/collapse icons, expand all/collapse all toolbar items and by using public methods. By default all tasks in Gantt was rendered in expanded state but we can change this status in Gantt.

Collapse all tasks at Gantt load

All tasks available in Gantt was rendered in collapsed state by setting [collapseAllParentTasks](#) property as **true**. The following code example shows how to use this property.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  collapseAllParentTasks: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="Gantt"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Define expand/collapse status of tasks

In Gantt, we can render some tasks in collapsed state and some tasks in expanded state, this can be done by defining expand status of the task in data source. This value was mapped to Gantt control by using [expandState](#) property. The following code example shows how to use this property.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    isExpand: true,
    subtasks: [
      { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
      { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
      { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
    ]
  },
  {
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    isExpand: false,
    subtasks: [
      { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
      { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
      { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 }
    ]
  },
];
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {

```



```

        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        expandState: 'isExpand',
        child: 'subtasks'
    }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customize expand/collapse action

On expand action [expanding](#) and [expanded](#) event will be triggered with current expanding row's information. Similarly on collapse action [collapsing](#) and [collapsed](#) event will be triggered. Using this events and it's arguments we can customize the expand/collapse action. The following code example shows how to prevent the particular row from expand/collapse action using [expanding](#) and [collapsing](#) event.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
```

```
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  collapsing: function (args: any) {
    if (args.data.TaskID == 1)
      args.cancel = true;
  },
  expanding: function (args: any) {
    if (args.data.TaskID == 5)
      args.cancel = true;
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Drag and drop

You can dynamically rearrange the rows in the Gantt control by using the `allowRowDragAndDrop` property. Using this property, row drag and drop can be enabled or disabled in Gantt. Using this feature, rows can be dropped at above and below as a sibling or child to the existing rows

To use row drag and drop feature, inject the `RowDD` and `Edit` module in Gantt.

INDEX.TS

```
import { Gantt, RowDD, Edit, Selection } from '@syncfusion/ej2-gantt';
import { projectNewData } from 'datasource.ts';
Gantt.Inject(RowDD, Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: projectNewData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  allowRowDragAndDrop: true
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
</body>
</html>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple row drag and drop

Gantt also supports dragging multiple rows at a time and drop them on any rows above, below, or at child positions. In Gantt, you can enable the multiple drag and drop by setting the `selectionSettings.type` to `Multiple` and you should enable the `allowRowDragAndDrop` property.

INDEX.TS

```

import { Gantt, RowDD, Edit, Selection } from '@syncfusion/ej2-gantt';
import { projectNewData } from 'datasource.ts';
Gantt.Inject(RowDD, Edit, Selection);
let gantt: Gantt = new Gantt({
    dataSource: projectNewData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    selectionSettings: {
        type: 'Multiple'
    },
    allowRowDragAndDrop: true
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Taskbar drag and drop between rows

The Gantt feature empowers users to efficiently reorganize records by seamlessly moving taskbar and rearranging their positions through a simple drag-and-drop action. Using this feature, rows can be dropped at above and below as a sibling or child to the existing rows.

This mode can be enable by setting the [allowTaskbarDragAndDrop](#) property to **true**.

To use row drag and drop feature, inject the **RowDD** and **Edit** module in Gantt.

INDEX.TS

```

import { Gantt, RowDD, Edit, Selection } from '@syncfusion/ej2-gantt';
import { projectNewData } from 'datasource.ts';
Gantt.Inject(RowDD, Edit, Selection);
let gantt: Gantt = new Gantt({
    dataSource: projectNewData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    allowTaskbarDragAndDrop: true,
    allowRowDragAndDrop: true
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop events

We provide various events to customize the row drag and drop action, the following table explains about the available events and its details.

Event Name | Description

rowDragStartHelper | Triggers when clicking the drag icon or Gantt row.

rowDragStart | Triggers when drag action starts in Gantt.

rowDrag | Triggers while dragging the Gantt row.

rowDrop | Triggers when a drag row was dropped on the target row.

Customize row drag and drop action

In Gantt, the **rowDragStartHelper** and **rowDrop** events are triggered on row drag and drop action. Using this event, you can prevent dragging of particular record, validate the drop position, and cancel the drop action based on the target record and dragged record. The following topics explain about this.

Prevent dragging of particular record

You can prevent drag action of the particular record by setting the **cancel** property to **true**, which is available in the **rowDragStartHelper** event argument based on our requirement. In the following sample, drag action was restricted for first parent record and its child records.

INDEX.TS

```
import { Gantt, RowDD, Edit, Selection } from '@syncfusion/ej2-gantt';
import { projectNewData } from 'datasource.ts';
Gantt.Inject(RowDD, Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: projectNewData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  rowDragStartHelper: function(args) {
    let record = args.data[0] ? args.data[0] : args.data;
    let taskId = record.ganttProperties.taskId;
    if (taskId <= 4) {
      args.cancel = true;
    }
  },
  allowRowDragAndDrop: true
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Validating drop position

You can prevent drop action based on the drop position and target record, by this, you can prevent dropping particular task on a specific task or specific position. This can be achieved by setting the `cancel` property to `true`, which is available in the `rowDrop` event argument.

In the following sample, we have prevented the drop action based on the position. In this sample, you cannot drop row as child in any of the available rows.

INDEX.TS

```

import { Gantt, RowDD, Edit, Selection } from '@syncfusion/ej2-gantt';
import { projectNewData } from 'datasource.ts';
Gantt.Inject(RowDD, Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: projectNewData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  rowDrop: function(args) {
    if (args.dropPosition == "middleSegment") {
      args.cancel = true;
    }
  },
  allowRowDragAndDrop: true
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```



```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent reordering a row as child to another row

You can prevent the default behavior of dropping rows as children to the target by setting the **cancel** property to **true** in [rowDrop](#) event argument. You can also change the drop position after cancelling using [reorderRows](#) method.

In the below example drop action is cancelled and dropped above to target row.

INDEX.TS

```

import { Gantt, RowDD, Edit, Selection } from '@syncfusion/ej2-gantt';
import { projectNewData } from 'datasource.ts';
Gantt.Inject(RowDD, Edit, Selection);
let gantt: Gantt = new Gantt({
    dataSource: projectNewData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    allowRowDragAndDrop: true,
    rowDrop : (args) => {
        if (args.dropPosition == 'middleSegment') {
            args.cancel = true;
            gantt.reorderRows([args.fromIndex], args.dropIndex, 'above');
        }
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Perform row drag and drop action programmatically

Gantt provides option to perform row drag and drop action programmatically by using the `reorderRows` method, this method can be used for any external actions like button click. The following arguments are used to specify the positions to drag and drop a row:

- `fromIndexes`: Index value of source(dragging) row.
- `toIndex`: Value of target index.
- `position`: Drop positions such as above, below, or child.

The following code example shows how to drag and drop a row on button click action.

INDEX.TS

```

import { Gantt, RowDD, Edit, Selection } from '@syncfusion/ej2-gantt';
import { projectNewData } from 'datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
Gantt.Inject(RowDD, Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: projectNewData,
  height: '450px',
  taskFields: {
    id: 'TaskID',

```

```

        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    allowRowDragAndDrop: true
});
gantt.appendTo('#Gantt');
let dragBtn: Button = new Button();
dragBtn.appendTo('#dynamicDrag');
document.getElementById('dynamicDrag').addEventListener('click', () => {
    gantt.reorderRows([1, 2, 3], 4, 'child');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="dynamicDrag">Drop records as child</button>
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize rows

You can customize the appearance of a row in grid side, by using the [rowDataBound](#) event and in chart side by using [queryTaskbarInfo](#) event

INDEX.TS

```
import { Gantt, RowDataBoundEventArgs, IQueryTaskbarInfoEventArgs } from
'@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  rowDataBound: rowBound,
  queryTaskbarInfo: queryTaskbarInfo,
});
gantt.appendTo('#Gantt');
function rowBound(args: RowDataBoundEventArgs) {
  if (args.data['TaskID'] == 4) {
    args.row.style.background = 'cyan';
  }
}
function queryTaskbarInfo(args: IQueryTaskbarInfoEventArgs) {
  if (args.data['TaskID'] == 4) {
    args.rowElement.style.background = 'cyan';
  }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Styling alternate rows

You can change the background colour of alternative rows in Gantt chart, by overriding the class as shown below.

```
.e-altrow, tr.e-chart-row:nth-child(even) {
background-color: #f2f2f2;
```

INDEX.TS

```
import { Gantt, RowDataBoundEventArgs, IQueryTaskbarInfoEventArgs } from
 '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <style>
        .e-altrow, tr.e-chart-row:nth-child(even) {
```

```

        background-color: #f2f2f2;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Row spanning

Gantt chart has an option to span row cells. You can achieve this using [rowSpan](#) attribute to span cells in the [QueryCellInfo](#) event.

In the following demo, **Soil test approval** cell is spanned to two rows in the **TaskName** column.

INDEX.TS

```

import { Gantt, QueryCellInfoEventArgs } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    gridLines: 'Both',
    queryCellInfo: queryCellInfo,
});
gantt.appendTo('#Gantt');
function queryCellInfo(args: QueryCellInfoEventArgs): void {
    if (args.data['TaskID'] == 4 && args.column.field === 'TaskName') {
        args.rowSpan = 2;
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Scrolling in EJ2 JavaScript Gantt control

The scrollbar will be displayed in the gantt when content exceeds the element **width** or **height**. The vertical and horizontal scrollbars will be displayed based on the following criteria:

- The vertical scrollbar appears when the total height of rows present in the gantt exceeds its element height.
- The horizontal scrollbar appears when the sum of columns width exceeds the grid pane size.
- The [height](#) and [width](#) are used to set the gantt height and width, respectively.

The default value for **height** and **width** is **auto**.

Set width and height

We can even set pixel values to width and height of gantt container using [width](#) and [height](#) properties.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,

```

```

height: '350px',
width: '600px',
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
},
editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
},
});
ganttt.appendTo('#Gantt')

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


Responsive with the parent container

Specify the [width](#) and [height](#) as **100%** to make the gantt element fill its parent container.

Setting the **height** to **100%** requires the gantt parent element to have explicit height. Also, the component will be responsive when the parent container is resized.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '100%',
  width: '100%',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  },
});
gantt.appendTo('#Gantt')
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-ganttresize {
      resize: both;
      overflow: auto;
      border: 1px solid red;
      padding: 10px;
      height: 300px;
      min-height: 250px;
      min-width: 250px;
    }
    .e-text{
```

```

        font-family: Helvetica, sans-serif;
        font-size: 14px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <p class="e-text"> The parent container can be resizable by dragging the
bottom-right corner.</p>
    <div id="container" class="e-ganttresize">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Scroll To Date method

In the Gantt control, When We use the [scrollToDate](#) method, it will scroll the timeline horizontally to the date that we specified in the method's argument.

The following code examples show how the scroll To Date method works in Gantt:

INDEX.TS

```

import { Gantt, Edit, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings: {
        position: "50%"
    }
    projectStartDate: new Date('04/01/2019'),
    projectEndDate: new Date('05/30/2019'),
});

```

```
gantt.appendTo('#Gantt');
let scrollBtn: Button = new Button();
scrollBtn.appendTo('#scroll');
document.getElementById('scroll').addEventListener('click', () => {
gantt.scrollToDate('05/27/2019');
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <button id="scroll">Scroll To Date</button>
  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Set the vertical scroll position

In the Gantt control, you can set the vertical scroller position dynamically by clicking the custom button using the [setScrollTop](#) method.

INDEX.TS

```
import { Gantt, Edit, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from 'datasource.ts';
Gantt.Inject(Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: data,
```

```

    height: '450px',
    taskFields: {
      id: 'TaskID',
      name: 'TaskName',
      startDate: 'StartDate',
      duration: 'Duration',
      progress: 'Progress',
      child: 'subtasks'
    },
    splitterSettings: {
      position: "50%"
    }
  });
gantt.appendTo('#Gantt');
let scrollBtn: Button = new Button();
scrollBtn.appendTo('#scrollTop');
document.getElementById('scrollTop').addEventListener('click', () => {
  gantt.ganttChartModule.scrollObject.setScrollTop(300);
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <button id="scrollTop">Change Scroll Position</button>
  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Managing Tasks

Managing tasks in EJ2 JavaScript Gantt control

The Gantt component has options to dynamically insert, delete, and update tasks in the project. The primary key column is necessary to manage the tasks and perform CRUD operations in Gantt. To define the primary key, set the [columns.isPrimaryKey](#) property to `true` in the particular column.

To use CRUD, inject the [Edit](#) module into the Gantt control.

INDEX.TS

```
import { Gantt, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  editSettings: {
    allowEditing: true,
    allowAdding: true,
    allowTaskbarEditing: true,
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Read-only Gantt

In Gantt, all create, update, delete operations can be disabled by set `readOnly` property as `true`. The following sample demonstrates, render Gantt chart as read only.

INDEX.TS

```

import { Gantt, Edit, Toolbar, ContextMenu, Resize, Sort, Selection } from
 '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, Toolbar, ContextMenu, Resize, Sort, Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    toolbar: ['Add', 'Cancel', 'CollapseAll', 'Delete', 'Edit', 'ExpandAll',
    'NextTimeSpan', 'PrevTimeSpan', 'Search', 'Update', 'Indent', 'Outdent'],
    enableContextMenu: true,
    editSettings: {
        allowEditing: true,
        allowAdding: true,
        allowDeleting: true,
        allowTaskbarEditing: true
    },
    allowSorting: true,
    allowResizing: true,
    readOnly: true
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<style>
.e-gantt-chart .e-preventEdit .e-right-resize-gripper,
.e-gantt-chart .e-preventEdit .e-left-resize-gripper,
.e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
.e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
.e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
    display: none;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Troubleshoot: Editing works only when primary key column is defined

Editing feature requires a primary key column for CRUD operations. While defining columns in Gantt using the [columns](#) property, it is mandatory that any one of the columns, must be a primary column. By default, the [id](#) column will be the primary key column. If [id](#) column is not defined, we need to enable [isPrimaryKey](#) for any one of the columns defined in the [columns](#) property.

Open new task dialog with default values

You can set default values when new task dialog opens using [actionBegin](#) event when `requestType` is `beforeOpenAddDialog`.

INDEX.TS

```

import { Gantt, Toolbar, Edit, ActionBeginArgs } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',

```

```

        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowAdding: true
    },
    actionBegin : (args: ActionBeginArgs) => {
        if (args.requestType == 'beforeOpenAddDialog') {
            args.rowData.TaskName = 'Gantt';
            args.rowData.Progress = 70;
            args.rowData.ganttProperties.taskName = 'Gantt';
            args.rowData.ganttProperties.progress = 70;
        }
    },
    toolbar: ['Add']
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```


Adding new tasks in EJ2 JavaScript Gantt control

Tasks can be dynamically added to the Gantt project by enabling the [editSettings.allowAdding](#) property.

Toolbar

A row can be added to the Gantt component from the toolbar while the [editSettings.allowAdding](#) property is set to true. On clicking the toolbar add icon, you should provide the task information in the add dialog.

INDEX.TS

```
import { Gantt, Toolbar, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  editSettings: {
    allowAdding: true
  },
  toolbar: ['Add']
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

By default, the new row will be added to the top most row in the Gantt control.

Context menu

A row can also be added above, below or child of the selected row by using context menu support. For this, we need to enable the property [enableContextMenu](#) and inject the [ContextMenu](#) module into the Gantt control.

INDEX.TS

```
import { Gantt, Edit, ContextMenu, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, ContextMenu, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  enableContextMenu: true,
  editSettings: {
    allowAdding: true
  },
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using method

You can add rows to the Gantt control dynamically using the [addRecord](#) method and you can define the add position of the default new record by using the [rowPosition](#) property. You can also pass the `rowIndex` as an additional parameter.

- Top of all the rows.
- Bottom to all the existing rows.
- Above the selected row.
- Below the selected row.
- As child to the selected row.

INDEX.TS

```

import { Gantt, Toolbar, Edit } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowAdding: true
    }
});
gantt.appendTo('#Gantt');
let addBtn: Button = new Button();
addBtn.appendTo('#addRow');
document.getElementById('addRow').addEventListener('click', () => {

```

```

    let record: Object = {
        TaskID: 10,
        TaskName: 'Identify Site location',
        StartDate: new Date('04/02/2019'),
        Duration: 3,
        Progress: 50
    };
    gantt.editModule.addRecord(record, 'Below', 2);
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="addRow">Add Row</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Editing tasks in EJ2 JavaScript Gantt control

The editing feature can be enabled in the Gantt control by enabling the [editSettings.allowEditing](#) and [editSettings.allowTaskbarEditing](#) properties.

The following editing options are available to update the tasks in the Gantt chart:

- Cell
- Dialog
- Taskbar

- Dependency links

Cell editing

By setting the edit mode to auto using the [editSettings.mode](#) property, the tasks can be edited through TreeGrid cells by double-clicking.

Note: If the [Edit](#) module is not injected, you cannot edit the tasks through TreeGrid cells.

The following code example shows you how to enable the cell editing in Gantt control.

INDEX.TS

```
import { Gantt, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  editSettings: {
    allowEditing: true,
    mode: 'Auto'
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

        <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: When the edit mode is set to **Auto**, on performing double-click action on TreeGrid side, the cells will be changed to editable mode and on performing double-click action on chart side, the edit dialog will appear for editing the task details.

Dialog editing

Modify the task details through the edit dialog by setting the edit [mode](#) to **Dialog**.

INDEX.TS

```

import { Gantt, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    editSettings: {
        allowEditing: true,
        mode: 'Dialog'
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: In dialog editing mode, the edit dialog appears when performing double-click action on both TreeGrid or Gantt chart sides.

Sections or tabs in dialog

In the Gantt dialog, you can define the required tabs or editing sections using the [addDialogFields](#) and [editDialogFields](#) properties. Every tab is defined using the [type](#) property.

INDEX.TS

```

import { Gantt, Edit, Toolbar } from '@syncfusion/ej2-gantt';
let ProjectResources: Object[] = [
    { resourceId: 1, resourceName: 'Martin Tamer' },
    { resourceId: 2, resourceName: 'Rose Fuller' },
    { resourceId: 3, resourceName: 'Margaret Buchanan' },
    { resourceId: 4, resourceName: 'Fuller King' },
    { resourceId: 5, resourceName: 'Davolio Fuller' },
    { resourceId: 6, resourceName: 'Van Jack' },
    { resourceId: 7, resourceName: 'Fuller Buchanan' },
    { resourceId: 8, resourceName: 'Jack Davolio' },
    { resourceId: 9, resourceName: 'Tamer Vinet' },
    { resourceId: 10, resourceName: 'Vinet Fuller' },
    { resourceId: 11, resourceName: 'Bergs Anton' },
    { resourceId: 12, resourceName: 'Construction Supervisor' }
];
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate: new
Date('04/02/2019'), Duration: 0, Progress: 50, info: 'Measure the total
property area allotted for construction' },
            {

```

```

        TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, resources: [2, 3, 5], info:
'Obtain an engineered soil test of lot where construction is planned.' +
    'From an engineer or company specializing in soil
testing'
    },
    { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Predecessor: "2FS", Progress: 50 },
    ]
},
{
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
        {
            TaskID: 6, TaskName: 'Develop floor plan for estimation',
            StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50, resources:
[4],
            isParent: false, info: 'Develop floor plans and obtain a
materials list for estimations'
        },
        { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50, resources: [4, 8], info: ''
},
        { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0, Predecessor: "6SS", Progress: 50,
resources: [12, 5], info: '' }
    ]
},
];
Gantt.Inject(Edit, Toolbar);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        resourceInfo: 'resources',
        duration: 'Duration',
        progress: 'Progress',
        notes: 'info',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    toolbar: ['Add', 'Edit', 'Delete', 'Cancel'],
    editDialogFields: [
        { type: 'General', headerText: 'General' },
        { type: 'Dependency' },
        { type: 'Resources' },
        { type: 'Notes' }
    ],
    addDialogFields: [
        { type: 'General', headerText: 'General' },
        { type: 'Dependency' }
    ]
});

```



```

    ],
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
    },
    resources: ProjectResources,
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        mode: 'Dialog',
        allowTaskbarEditing: true
    }
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt-chart .e-preventEdit .e-right-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
    .e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
      display: none;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Limiting data fields in general tab

In the Gantt dialog, you can make only specific data source fields visible for editing by using the [addDialogFields](#) and [editDialogFields](#) properties. The data fields are defined with [type](#) and [fields](#) properties.

Note: You can also define the custom fields in the add/edit dialog General tab using the [fields](#) property.

INDEX.TS

```
import { Gantt, Edit, Toolbar } from '@syncfusion/ej2-gantt';
let ProjectResources: Object[] = [
  { resourceId: 1, resourceName: 'Martin Tamer' },
  { resourceId: 2, resourceName: 'Rose Fuller' },
  { resourceId: 3, resourceName: 'Margaret Buchanan' },
  { resourceId: 4, resourceName: 'Fuller King' },
  { resourceId: 5, resourceName: 'Davolio Fuller' },
  { resourceId: 6, resourceName: 'Van Jack' },
  { resourceId: 7, resourceName: 'Fuller Buchanan' },
  { resourceId: 8, resourceName: 'Jack Davolio' },
  { resourceId: 9, resourceName: 'Tamer Vinet' },
  { resourceId: 10, resourceName: 'Vinet Fuller' },
  { resourceId: 11, resourceName: 'Bergs Anton' },
  { resourceId: 12, resourceName: 'Construction Supervisor' }
];
let GanttData: Object[] = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    isParent: true,
    subtasks: [
      { TaskID: 2, TaskName: 'Identify Site location', StartDate: new
Date('04/02/2019'), Duration: 0, Progress: 50, isParent: false, info:
'Measure the total property area allotted for construction' },
      {
        TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50, resources: [2, 3, 5],
isParent: false, info: 'Obtain an engineered soil test of lot where
construction is planned.' +
        'From an engineer or company specializing in soil
testing'
      },
      { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Predecessor: "2FS", Progress: 50, isParent:
false },
    ]
  },
  {
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    isParent: true,
```

```

        subtasks: [
            {
                TaskID: 6, TaskName: 'Develop floor plan for estimation',
                StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50, resources:
                [4],
                isParent: false, info: 'Develop floor plans and obtain a
                materials list for estimations'
            },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
            Date('04/04/2019'), Duration: 3, Progress: 50, resources: [4, 8], isParent:
            false, info: '' },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
            Date('04/04/2019'), Duration: 0, Predecessor: "6SS", Progress: 50,
            resources: [12, 5], isParent: false, info: '' }
        ]
    },
];
Gantt.Inject(Edit, Toolbar);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        resourceInfo: 'resources',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    toolbar: ['Add', 'Edit', 'Delete', 'Cancel'],
    editDialogFields: [
        { type: 'General', headerText: 'General', fields: ['TaskID',
        'TaskName', 'isParent'] },
        { type: 'Dependency' },
        { type: 'Resources' }
    ],
    addDialogFields: [
        { type: 'General', headerText: 'General', fields: ['TaskID',
        'TaskName', 'isParent'] },
        { type: 'Resources' },
        { type: 'Dependency' }
    ],
    columns: [
        { field: 'TaskID', headerText: 'Task ID', width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width: '250' },
        { field: 'isParent', headerText: 'Custom Column', width: '100' },
        { field: 'resources', headerText: 'Resources', width: '200' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
    },
},

```

```

resources: ProjectResources,
editSettings: {
    allowAdding: true,
    allowEditing: true,
    mode: 'Dialog',
    allowTaskbarEditing: true
}
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt-chart .e-preventEdit .e-right-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
    .e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
      display: none;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Validation in EJ2 JavaScript Gantt control

Column validation

Column validation validates the editing and adding data and it display errors for invalid fields before saving data. This is effective in both inline and dialog editing.

Gantt uses [Form Validator](#) component for column validation. You can set [validation rules](#) by defining the [columns.validationRules](#). The value cannot be saved unless the validation rule get satisfied.

INDEX.TS

```
import { Gantt, Edit, Selection, Toolbar } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, Selection, Toolbar);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel'],
  columns: [
    { field: 'TaskID' },
    { field: 'TaskName', validationRules: { required: true } },
    { field: 'StartDate', editType: 'datetimepickeredit', edit: { params:
{ format: 'M/d/y hh:mm a' } },
      format: { format: 'M/d/y hh:mm a', type: 'dateTime' },
      validationRules: { required: true, date: true } },
    { field: 'Duration', validationRules: { required: true } },
    { field: 'Progress', validationRules: { required: true } }
  ],
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom validation

You can define your own custom validation rules for the specific columns by using callback function to it's [validation rule](#).

In the below demo, custom validation applied for **TaskName** column.

INDEX.TS

```

import { Gantt, Edit, Selection, Toolbar } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, Selection, Toolbar);
let customFn: (args: { [key: string]: string }) => boolean = (args: { [key:
string]: string }) => {
    return args['value'].length >= 8;
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel'],
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', validationRules: { required: true, minLength:
[customFn, 'Value should be greater than 8 letters'] } },
        { field: 'StartDate', editType: 'datetimepickeredit', edit: { params:
{ format: 'M/d/y hh:mm a' } } },

```

```

        format: { format: 'M/d/y hh:mm a', type: 'dateTime' },
validationRules: { required: true, date: true } },
        { field: 'Duration', validationRules: { required: true } },
        { field: 'Progress', validationRules: { required: true } }
    ],
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dependency and resource grid validation

Validation rules can also be implemented for the dependency and resource grid in the add or edit dialog by employing the event [actionBegin](#).

Within the `actionBegin` event, `validationRules` can be configured for columns in the grid of the dependency and resource tabs using the `requestType` `beforeOpenEditDialog` or `beforeOpenAddDialog`.

INDEX.TS

```
import { Gantt, Edit, Selection, Toolbar } from '@syncfusion/ej2-gantt';
import { GanttData, editingResources } from 'datasource.ts';
Gantt.Inject(Edit, Selection, Toolbar);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    dependency: 'Predecessor',
    progress: 'Progress',
    child: 'subtasks',
    resourceInfo: 'resources'
  },
  resourceFields: {
    id: 'resourceId',
    name: 'resourceName'
  },
  resources: editingResources,
  actionBegin: function (args: any) {
    if (args.requestType == "beforeOpenEditDialog" || args.requestType == "beforeOpenAddDialog") {
      args.Dependency.columns[3].validationRules = { required: true };
      args.Resources.columns[2].allowEditing = true;
      args.Resources.columns[2].validationRules = { required: true };
    }
  },
  toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel'],
  columns: [
    { field: 'TaskID' },
    { field: 'TaskName', validationRules: { required: true } },
    { field: 'StartDate', editType: 'datetimepickeredit', edit: { params: { format: 'M/d/y hh:mm a' } },
      format: { format: 'M/d/y hh:mm a', type: 'dateTime' } },
    { field: 'Duration', validationRules: { required: true } },
    { field: 'Progress', validationRules: { required: true } }
  ],
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    allowTaskbarEditing: true,
    showDeleteConfirmDialog: true
  }
});
gantt.appendTo('#Gantt');
```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Deleting tasks in EJ2 JavaScript Gantt control

A task delete option in the Gantt control can be enabled by enabling the [ediSettings.allowDeleting](#) property. Tasks can be deleted by clicking the delete toolbar item or using the `deleteRow` method. You can call this method dynamically on any custom actions like button click. The following code example shows how to enable the delete option in the Gantt control.

INDEX.TS

```

import { Gantt, Edit, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from '@syncfusion/ej2-datasource';
import { getValue } from '@syncfusion/ej2-base';
Gantt.Inject(Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',

```

```

        child: 'subtasks'
      },
      editSettings: {
        allowDeleting: true
      }
    });
    gantt.appendTo('#Gantt');
    let delBtn: Button = new Button();
    delBtn.appendTo('#deleteRecord');
    document.getElementById('deleteRecord').addEventListener('click', () => {
      gantt.editModule.deleteRecord(getValue('TaskID',
      gantt.selectionModule.getSelectedRecords()[0]));
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <button id="deleteRecord">Delete Record</button>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

NOTE: You should select any one of the rows in the Gantt control to perform task delete action.

You should set the [allowDeleting](#) value to `true` to delete the record dynamically.

Delete confirmation message

Delete confirmation message is used to get the confirmation from users before deleting a task. This confirmation message can be enabled by setting the [editSettings.showDeleteConfirmDialog](#) property to true.

The following code snippet explains how to enable the delete confirmation message in Gantt.

INDEX.TS

```
import { Gantt, Edit, Toolbar, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, Toolbar, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  toolbar: ['Delete'],
  editSettings: {
    allowDeleting: true,
    showDeleteConfirmDialog: true
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt-chart .e-preventEdit .e-right-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
    .e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
      display: none;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Task bar editing in EJ2 JavaScript Gantt control

Modify the task details through user interaction such as resizing and dragging the taskbar by enabling the [allowTaskbarEditing](#) property.

INDEX.TS

```

import { Gantt, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowTaskbarEditing: true
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent editing for specific tasks

On taskbar edit action, the [taskbarEditing](#) event will be triggered. You can prevent the taskbar from editing using the [taskbarEditing](#) event. This can be done by setting cancel property of [taskbarEditing](#) event argument to true. And we can hide the taskbar editing indicators like taskbar resizer, progress resizer and connector points by using [queryTaskbarInfo](#) event. The following code example shows how to achieve this.

INDEX.TS

```

import { Gantt, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    taskbarEditing: function (args: any) {
        if (args.data.TaskID == 4) // We can't edit Task Id 4
            args.cancel = true;
    },
    queryTaskbarInfo: function (args) {
        if (args.data.TaskID == 6) {
            args.taskbarElement.className += ' e-preventEdit' // Taskbar
            editing indicators are hidden
        }
    }
}

```

```

    },
    editSettings: {
        allowTaskbarEditing: true
    }
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
    .e-gantt-chart .e-preventEdit .e-right-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
    .e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
      display: none;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Task dependencies

In the Gantt control, you can update the dependencies between the tasks and link the tasks interactively. The task dependencies can be mapped from the data source using the [dependency](#) property.

You can update the task dependencies using the following ways:

- Mouse interactions: Using connector points in the taskbar, you can perform drag and drop action to create task dependency links.
- Edit dialog: Create or remove the task dependencies using the **Dependency** tab in the edit dialog.
- Cell editing: Create or remove the task links using cell editing.

The following code example demonstrates how to enable task dependency editing in the Gantt chart using the [editSettings](#) property.

INDEX.TS

```
import { Gantt, Edit } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
  {
    TaskID: 1,
    TaskName: 'Project Initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    isParent: true,
    subtasks: [
      { TaskID: 2, TaskName: 'Identify Site location', StartDate: new
Date('04/02/2019'), Duration: 0, Progress: 50 },
      { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
      { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Predecessor: "2FS", Progress: 50 },
    ]
  },
  {
    TaskID: 5,
    TaskName: 'Project Estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    isParent: true,
    subtasks: [
      { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
      { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
      { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 0, Predecessor: "6SS", Progress: 50 }
    ]
  },
];
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
```

```

        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    editSettings: {
        allowTaskbarEditing: true,
        allowEditing: true,
        mode: 'Auto'
    }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

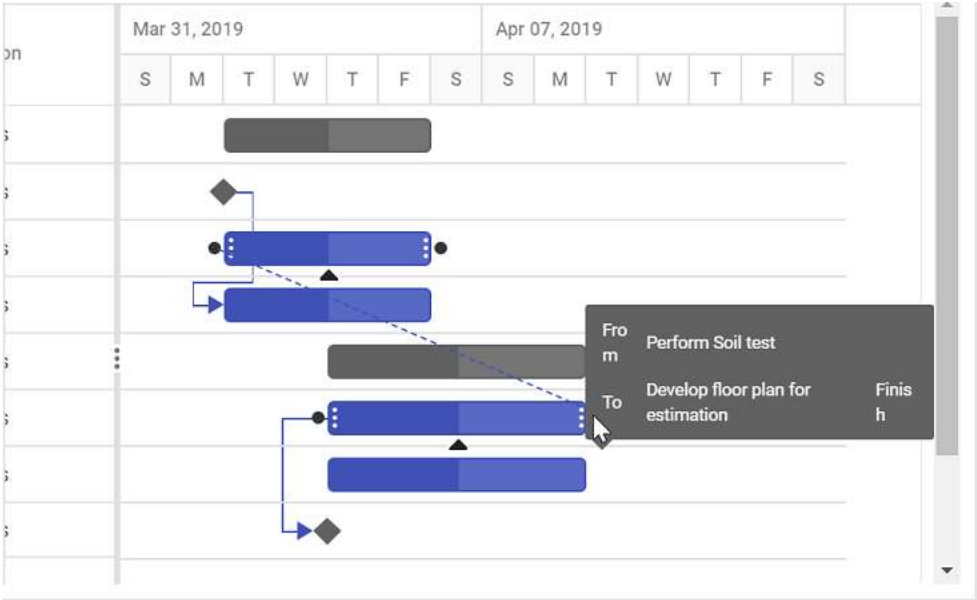
```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

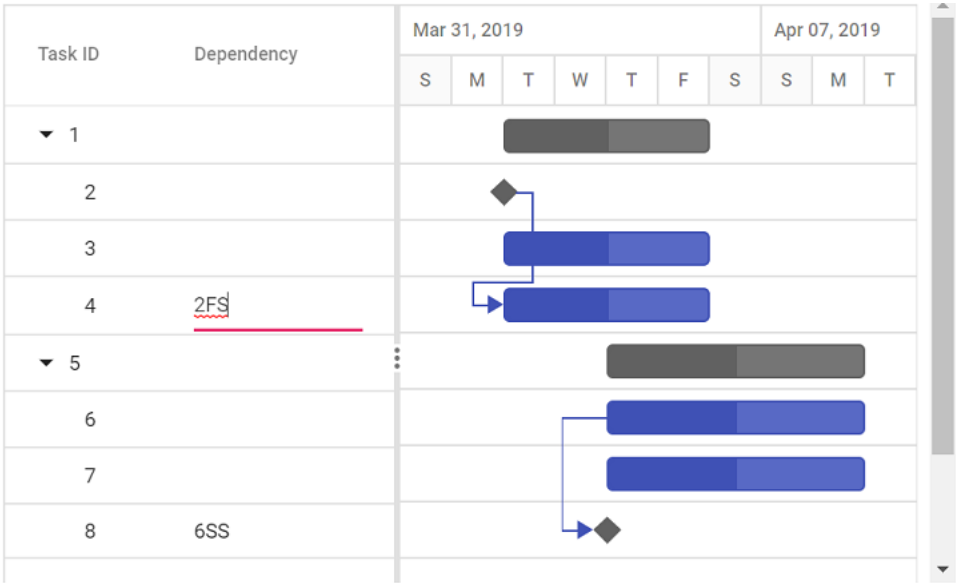
  <style>
    .e-gantt-chart .e-preventEdit .e-right-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-progress-resize-gripper,
    .e-gantt-chart .e-preventEdit .e-left-connectorpoint-outer-div,
    .e-gantt-chart .e-preventEdit .e-right-connectorpoint-outer-div {
      display: none;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Updating with mouse interaction action



Updating with cell edit

Task Information

×

GENERAL

DEPENDENCY

+

Add

🗑️

Delete

ID	Name	Type	Offset
2	2-Identify Site location	Finish-Start	0 days

SAVE

CANCEL

Updating with dialog

Note: When the edit mode is set to **Auto**, on performing double-click action on TreeGrid side, the cells will be changed to editable mode and on performing double-click action on chart side, the edit dialog will appear for editing the task details.

Update task values using method

Tasks' value can be dynamically updated by using the [updateRecordById](#) method. You can call this method on any custom action. The following code example shows how to use this method to update a task.

NOTE: Using the [updateRecordById](#) method, you cannot update the task ID value.

INDEX.TS

```
import { Gantt, Edit } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  editSettings: {
    allowEditing: true
  }
});
gantt.appendTo('#Gantt');
let updateBtn: Button = new Button();
updateBtn.appendTo('#updateRecord');
document.getElementById('updateRecord').addEventListener('click', () => {
```

```

let data: Object = {
    TaskID: 3,
    TaskName: 'Updated by index value',
    StartDate: new Date('04/02/2019'),
    Duration: 4,
    Progress: 50
};
ganttt.updateRecordByID(data);
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <button id="updateRecord">Update Record</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell edit type and its params

The [columns.editType](#) is used to define the edit type for any particular column.

You can set the [columns.editType](#) based on data type of the column.

- **numericedit** - [NumericTextBox](#) component for integers, double, and decimal data types.
- **defaultedit** - [TextBox](#) component for string data type.
- **dropdownedit** - [DropDownList](#) component to show all unique values related to that field.
- **booleanedit** - [CheckBox](#) component for boolean data type.
- **datepickeredit** - [DatePicker](#) component for date data type.

- **datetimepickeredit** - [DateTimePicker](#) component for date time data type.

Also, you can customize the behavior of the editor component through the [columns.edit.params](#).

The following table describes cell edit type component and their corresponding edit params of the column.

Edit Type | Component | Example

numericedit | [NumericTextBox](#) | params: { decimals: 2, value: 5 }

dropdownedit | [DropDownList](#) | params: { value: 'Germany' }

booleanedit | [Checkbox](#) | params: { checked: true }

datepickeredit | [DatePicker](#) | params: { format: 'dd.MM.yyyy' }

datetimepickeredit | [DateTimePicker](#) | params: { value: new Date() }

INDEX.TS

```
import { Gantt, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit);
function durationFormat(field: string, data: Object, column: Object): string
{
    return data[field];
}
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowEditing: true
    },
    toolbar: ['Edit'],
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        { field: 'TaskName', headerText: 'Task Name' },
        { field: 'StartDate', headerText: 'Start Date' },
        { field: 'Duration', headerText: 'Duration', editType: 'numericedit',
        edit: { params: { min: 1 } }, valueAccessor: durationFormat },
        { field: 'Progress', headerText: 'Progress', edit: { params: {
showSpinButton: false } } },
    ],
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell Edit Template

The cell edit template is used to create a custom component for a particular column by invoking the following functions:

- **create** - It is used to create the element at the time of initialization.
- **write** - It is used to create the custom component or assign default value at the time of editing.
- **read** - It is used to read the value from the component at the time of save.
- **destroy** - It is used to destroy the component.

INDEX.TS

```

import { Gantt, Edit } from '@syncfusion/ej2-gantt';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit);
let elem: HTMLElement;
let dropdownlistObj: DropDownList;
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',

```

```

        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowEditing: true
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        {
            field: 'TaskName', headerText: 'Task Name',
            edit: {
                create: () => {
                    elem = document.createElement('input');
                    return elem;
                },
                read: () => {
                    return dropdownlistObj.value;
                },
                destroy: () => {
                    dropdownlistObj.destroy();
                },
                write: (args: Object) => {
                    dropdownlistObj = new DropDownList({
                        dataSource: gantt.treeGrid.grid.dataSource,
                        fields: { value: 'TaskName' },
                        value: args.rowData[args.column.field],
                        floatLabelType: 'Auto',
                    });
                    dropdownlistObj.appendTo(elem);
                }
            }
        },
        { field: 'StartDate', headerText: 'Start Date' },
        { field: 'Duration', headerText: 'Duration' },
        { field: 'Progress', headerText: 'Progress' },
    ],
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable editing for particular column

You can disable editing for particular columns, by using the [columns.allowEditing](#) property.

In the following demo, editing is disabled for the **TaskName** column.

INDEX.TS

```

import { Gantt, Edit } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowEditing: true
    },
    toolbar: ['Edit'],
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        { field: 'TaskName', headerText: 'Task Name', allowEditing: false },
        { field: 'StartDate', headerText: 'Start Date' },
        { field: 'Duration', headerText: 'Duration' },
        { field: 'Progress', headerText: 'Progress' }
    ],
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

In dent and out dent in EJ2 JavaScript Gantt control

Indent and Outdent of a task are used to update the level of task in the hierarchical order of the task. It can be performed by enabling the [editSettings.allowEditing](#) property.

Indent - Selected task can be indented to the level of task to the hierarchical order. It can be performed by using in-built context menu or toolbar items. It can also be invoked by using the `indent` method dynamically on any action like external button click. The following code example shows how to enable indent option in the Gantt chart.

Outdent - Selected task can be outdented to the level of task from the hierarchical order. It can be performed by using in-built context menu or toolbar items. It can also be invoked by using the `outdent` method dynamically on any action like external button click. The following code example shows how to enable outdent option in the Gantt chart.

INDEX.TS

```

import { Gantt, Toolbar, Edit, Selection, Toolbar } from '@syncfusion/ej2-
gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit, Selection, Toolbar);

```



```

let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar: ['Indent', 'Outdent'],
    editSettings: {
        allowEditing: true
    }
});
gantt.appendTo('#Gantt');
let ind: Button = new Button();
ind.appendTo('#indent');
let out: Button = new Button();
out.appendTo('#outdent');
document.getElementById('indent').addEventListener('click', () => {
    gantt.indent();
});
document.getElementById('outdent').addEventListener('click', () => {
    gantt.outdent();
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="indent">Indent</button>
        <button id="outdent">Outdent</button>
        <div id="Gantt"></div>
    </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Splitting and merging tasks in EJ2 JavaScript Gantt control

Splitting task at load time

To split task at load time, we can define segment details in both hierarchical and self-referential way.

Refer below link for more details.

- [Split task at load time](#)

Split task dynamically

The task can be splitted dynamically, either by using the context menu or dialog.

- **Dialog:** Segments tab is rendered in add/edit dialog, when the [taskFields.segments](#) or [taskFields.segmentId](#) property is mapped. Using this tab, we can split the task based on the original start and end date of a particular task.
- **Context menu:** When the [taskFields.segments](#) or [taskFields.segmentId](#) property is mapped and the [enableContextMenu](#) property is enabled, Split Task item will be included in the context menu.

INDEX.TS

```

import { Gantt, Edit, Toolbar, Selection, ContextMenu } from
 '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, Toolbar, Selection, ContextMenu);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: "450px",
    taskFields: {
        id: "TaskID",
        name: "TaskName",
        startDate: "StartDate",
        endDate: "EndDate",
        duration: "Duration",
        progress: "Progress",
        child: "subtasks",
        segments: "Segments"
    },
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
 'CollapseAll'],
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    }
});

```

```

    },
    enableContextMenu: true
  });
  gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Merge tasks

The splitted tasks can be merged either by using the **Merge Task** item of the Context menu or by using the dialog. We can also merge the tasks, by simply dragging the segments together in the UI.

Limitations of Split tasks

1. Parent and milestone tasks cannot be split into segments.
2. The task must have a width greater than the timeline unit cell in order to be split.
3. Split task is not supported in the **Resource view**.

Maintaining data in server in EJ2 JavaScript Gantt control

All the modified data in Gantt control can be maintained in the database using RESTful web services.

All the CRUD operations in the gantt are done through DataManager. The DataManager has an option to bind all the CRUD related data in server-side.

In the below section, we have explained how to get the edited data details on the server-side using the `UrlAdaptor`.

URL Adaptor

In Gantt, we can fetch data from SQL database using `ADO.NET` Entity Data Model and update the changes on CRUD action to the server by using `DataManager` support. To communicate with the remote data we are using `UrlAdaptor` of `DataManager` property to call the server method and get back resultant data in JSON format. We can know more about `UrlAdaptor` from [here](#).

Please refer the [link](#) to create the `ADO.NET` Entity Data Model in Visual studio,

We can define data source for Gantt as instance of `DataManager` using `url` property of `DataManager`. Please Check the below code snippet to assign data source to Gantt.

```
`ts
import { Gantt } from '@syncfusion/ej2-gantt';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
let dataSource: DataManager = new DataManager({
url: '/Home/UrlDatasource',
adaptor: new UrlAdaptor
});
let gantt: Gantt = new Gantt({
dataSource: dataSource,
height: '450px',
treeColumnIndex: 1,
taskFields: {
id: 'TaskId',
name: 'TaskName',
startDate: 'StartDate',
progress: 'Progress',
duration: 'Duration',
dependency: 'Predecessor',
child: 'SubTasks'
}
});
gantt.appendTo('#Gantt');
`ts

GanttDataSourceEntities db = new GanttDataSourceEntities();
```

```

public ActionResult UrlDatasource(DataManagerRequest dm)
{
    List<GanttData>DataList = db.GanttDatas.ToList();
    var count = DataList.Count();
    return Json(new { result = DataList, count = count });
}

```

We can also do CRUD operations over Gantt data and save the changes to database. By using **BatchUrl** property of **DataManager**, we can communicate with the controller method to update the data source on CRUD operation. In gantt CRUD actions on task are dependent with other tasks. For example on editing the child record on chart side, corresponding parent item also will get affect and predecessor dependency task as well get affected. So in Gantt all the CRUD operations are considered to be batch editing where you will get all the affected records as collection. Please check the below code snippet to assign controller method to this property.

```

`ts
import { Gantt } from '@syncfusion/ej2-gantt';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';

let dataSource: DataManager = new DataManager({
    url: '/Home/UrlDatasource',
    adaptor: new UrlAdaptor,
    batchUrl: "Home/BatchSave"
});

let gantt: Gantt = new Gantt({
    dataSource: dataSource,
    height: '450px',
    treeColumnIndex: 1,
    taskFields: {
        id: 'TaskId',
        name: 'TaskName',
        startDate: 'StartDate',
        progress: 'Progress',
        duration: 'Duration',
        dependency: 'Predecessor',
        child: 'SubTasks'
    }
}

```

```

});
gantt.appendTo('#Gantt');
`ts
public class ICRUDModel<T> where T : class
{
    public object key { get; set; }
    public T value { get; set; }
    public List<T> added { get; set; }
    public List<T> changed { get; set; }
    public List<T> deleted { get; set; }
}
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult BatchSave([FromBody]ICRUDModel<GanttData> data)
{
    List<GanttData> uAdded = new List<GanttData>();
    List<GanttData> uChanged = new List<GanttData>();
    List<GanttData> uDeleted = new List<GanttData>();
    //...
    return Json(new { addedRecords = uAdded, changedRecords = uChanged, deletedRecords = uDeleted });
}
`ts

```

This server method will be triggered for all the CRUD operations like adding, editing and deleting actions. We can handle those each operations separately inside this method with corresponding data received in this method argument. Also, when using the `UrlAdaptor`, you need to return the data as JSON from the controller action and the JSON object must contain a property as result with dataSource as its value and one more property count with the dataSource total records count as its value.

Insert action

Using the `added` argument of the `BatchUrl` method we can insert the newly added row to database and return the same to client side. please find the below code example for details.

```

`ts
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult BatchSave([FromBody]ICRUDModel<GanttData> data)
{
    List<GanttData> uAdded = new List<GanttData>();

```

```
//Performing insert operation
if (data.added != null && data.added.Count() > 0)
{
    foreach (var rec in data.added)
    {
        uAdded.Add(this.Create(rec));
    }
}
return Json(new { addedRecords = uAdded });
//...
}

public GanttData Create(GanttData value)
{
    db.GanttDatas.Add(value);
    db.SaveChanges();
    return value;
}
`
```

Editing action

Using the `changed` argument of the `BatchUrl` method we can update the modified records to database and return the same to client side. please find the below code example for details.

```
`ts
GanttDataSourceEntities db = new GanttDataSourceEntities();
public ActionResult BatchSave([FromBody]ICRUDModel<GanttData> data)
{
    List<GanttData> uChanged = new List<GanttData>();
    //Performing update operation
    if (data.changed != null && data.changed.Count() > 0)
    {
        foreach (var rec in data.changed)
        {
            uChanged.Add(this.Edit(rec));
        }
    }
}
```

```

return Json(new { changedRecords = uChanged });
//...
}

public GanttData Edit(GanttData value)
{
    GanttData result = db.GanttDatas.Where(currentData => currentData.TaskId ==
value.TaskId).FirstOrDefault();
    if (result != null)
    {
        result.TaskId = value.TaskId;
        result.TaskName = value.TaskName;
        result.StartDate = value.StartDate;
        result.EndDate = value.EndDate;
        result.Duration = value.Duration;
        result.Progress = value.Progress;
        result.Predecessor = value.Predecessor;
        db.SaveChanges();
        return result;
    }
    else
    {
        return null;
    }
}

```

Delete action

Using the `deleted` argument of the `BatchUrl` method we can remove the deleted records from database and return the same to client side. on deleting the record we need to remove its corresponding child records as well if it exist from the data base. please find the below code example for details.

```

`ts

GanttDataSourceEntities db = new GanttDataSourceEntities();

public ActionResult BatchSave([FromBody]ICRUDModel<GanttData> data)
{
    List<GanttData> uDeleted = new List<GanttData>();

```



```
//Performing delete operation
if (data.deleted != null && data.deleted.Count() > 0)
{
    foreach (var rec in data.deleted)
    {
        uDeleted.Add(this.Delete(rec.TaskId));
    }
}
return Json(new { deletedRecords = uDeleted });
}

public GanttData Delete(string value)
{
    var result = db.GanttDatas.Where(currentData => currentData.TaskId == value).FirstOrDefault();
    db.GanttDatas.Remove(result);
    RemoveChildRecords(value);
    db.SaveChanges();
    return result;
}

public void RemoveChildRecords(string key)
{
    var childList = db.GanttDatas.Where(x => x.ParentId == key).ToList();
    foreach (var item in childList)
    {
        db.GanttDatas.Remove(item);
        RemoveChildRecords(item.TaskId);
    }
}
}
```

Columns

Columns in EJ2 JavaScript Gantt control

The column displays information from a bound data source, and you can edit the values of column to update the task details through TreeGrid. The operations such as sorting, filtering, and searching can be performed based on column definitions. To display a Gantt column, the [field](#) property should be mapped from the data source to the column.

If the column [field](#) is not specified in the data source, the column values will be empty.

The [treeColumnIndex](#) property is used to define the expander column in the Gantt control to expand and collapse the child rows.

Defining columns

Using the [columns](#) property, you can define the columns in Gantt. If the columns are not defined, then the default columns will be rendered based on the mapped data source fields in the [taskFields](#) property. Refer to the following code example for defining the columns in Gantt along with their widths.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 }
        ]
    },
];
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    height: '450px',
    splitterSettings: {
        columnIndex: 2
    },
});
```

```

        columns: [
            { field: 'TaskID', width: '150' },
            { field: 'TaskName', width: '250' }
        ]
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom column header

The column header text can be defined using the [headerText](#) property, and you can customize the column headers using the [headerTemplate](#) property.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',

```

```

        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    height: '450px',
    splitterSettings: {
        columnIndex: 4
    },
    columns: [
        { field: 'TaskName', headerTemplate: '#projectName', width: '150' },
        { field: 'StartDate', headerTemplate: '#dateTemplate', width: '150' },
        { field: 'Duration', headerTemplate: '#durationTemplate',
headerText: 'Duration', width: '150' },
        { field: 'Progress', headerTemplate: '#progressTemplate',
headerText: 'Progress', width: '150' },
    ]
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>

    <script type="text/x-template" id="projectName">
        <div>
            <div>
                 Task Name
            </div>
        </div>
    </script>
    <script type="text/x-template" id="dateTemplate">

```

```

        <div>
            <div>
                 Start Date
            </div>
        </div>
    </script>
    <script type="text/x-template" id="durationTemplate">
        <div>
            <div>
                 Duration
            </div>
        </div>
    </script>
    <script type="text/x-template" id="progressTemplate">
        <div>
            <div>
                 Progress
            </div>
        </div>
    </script>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Format

To format the cell values based on a specific culture, use the [columns.format](#) property. The Gantt control uses the [Internationalization](#) library to format **number** and **date** values.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings: {
        columnIndex: 3
    },
    height: '450px',
    columns: [

```

```

    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
    { field: 'Progress', headerText: 'Progress', width: '150', format:
'C' },
    { field: 'TaskName', headerText: 'Task Name', width: '150' },
    { field: 'StartDate', headerText: 'Start Date', width: '150' },
    { field: 'Duration', headerText: 'Duration', width: '150' }
  ]
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the number and date values are formatted in **en-US** culture.

Number formatting

The number or integer values can be formatted using the following format strings.

Format	Description	Remarks
{ type:'date', format:'dd/MM/yyyy' }	04/07/2019	
{ type:'date', format:'dd.MM/yyyy' }	04.07.2019	

```
{ type:'date', skeleton:'short' } | 7/4/19
```

```
{ type: 'dateTime', format: 'dd/MM/yyyy hh:mm a' } | 04/07/2019 12:00 AM
```

```
{ type: 'dateTime', format: 'MM/dd/yyyy hh:mm:ss a' } | 07/04/2019 12:00:00 AM
```

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  splitterSettings: {
    columnIndex: 3
  },
  height: '450px',
  columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
    { field: 'TaskName', headerText: 'Task Name', width: '150' },
    { field: 'Progress', headerText: 'Progress', width: '150', format:
'C' },
    { field: 'StartDate', headerText: 'Start Date', width: '150',
format: { format: 'dd/MM/yyyy', type: 'date' } },
    { field: 'Duration', headerText: 'Duration', width: '150' }
  ]
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
      <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column reordering in EJ2 JavaScript Gantt control

The column reordering can be done by dragging a column header from one index to another index within the TreeGrid. To enable reordering, set the [allowReordering](#) property to true.

To use the column reordering feature, inject the **Reorder** module to the Gantt control.

INDEX.TS

```

import { Gantt, Reorder } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Reorder);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  allowReordering: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  height: '450px',
  splitterSettings: {
    columnIndex: 5
  }
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can disable the reordering of a particular column by setting the [columns.allowReordering](#) property to false.

Reorder multiple columns

Multiple columns can be reordered at a time by using the [reorderColumns](#) method.

INDEX.TS

```

import { Gantt, Reorder } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Reorder);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    allowReordering: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    height: '450px',
    splitterSettings: {
        position: '100%'
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' }
    ]
}

```

```
});
gantt.appendTo('#Gantt');
let reorderMultipleCols: Button = new Button();
reorderMultipleCols.appendTo('#reorderMultipleCols');
document.getElementById('reorderMultipleCols').addEventListener('click', ()
=> {
    gantt.reorderColumns('TaskName', 'Progress');
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="reorderMultipleCols">Reorder Task ID and Task Name to
Last</button>
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Reorder Events

During the reorder action, the gantt component triggers the below three events.

1. The [columnDragStart](#) event triggers when column header element drag (move) starts.
2. The [columnDrag](#) event triggers when column header element is dragged (moved) continuously.
3. The [columnDrop](#) event triggers when a column header element is dropped on the target column.

INDEX.TS

```
import { Gantt, Reorder } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Reorder);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  allowReordering: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  splitterSettings: {
    columnIndex: 4
  },
  height: '450px',
  columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
    { field: 'TaskName', headerText: 'Task Name', width: '150' },
    { field: 'StartDate', headerText: 'Start Date', width: '150' },
    { field: 'Duration', headerText: 'Duration', width: '150' },
    { field: 'Progress', headerText: 'Progress', width: '150' }
  ],
  columnDragStart: () => {
    alert('columnDragStart event is Triggered');
  },
  columnDrag: () => {
    alert('columnDrag event is Triggered');
  },
  columnDrop: () => {
    alert('columnDrop event is Triggered');
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column resizing in EJ2 JavaScript Gantt control

The column width can be resized by clicking and dragging the right edge of the column header. While dragging, the width of the column will be resized immediately. Each column can be auto resized by double-clicking the right edge of the column header to fit the width of that column based on the widest cell content. To resize the column, set the [allowResizing](#) property to true. The following code example shows how to enable the column resize feature in the Gantt control.

To resize the column, inject the **Resize** module into the Gantt control.

INDEX.TS

```

import { Gantt, Resize } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Resize);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    allowResizing: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings: {
        columnIndex: 4
    },
    height: '450px'
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can disable resizing for a particular column by setting the [columns.allowResizing](#) to `false`.

Defining minimum and maximum column width

The column resizing can be restricted between minimum and maximum widths by defining the [columns->minWidth](#) and [columns->maxWidth](#) properties.

In the following example, the minimum and maximum widths are defined for the **Duration**, and **Task Name** columns.

INDEX.TS

```

import { Gantt, Resize } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Resize);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  allowResizing: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  splitterSettings: {
    columnIndex: 5
  },
  height: '450px',

```

```

        columns: [
            { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
            { field: 'TaskName', headerText: 'Task Name', width: '200',
minWidth: '150', maxWidth: '250', },
            { field: 'StartDate', headerText: 'Start Date', width: '150' },
            { field: 'Duration', headerText: 'Duration', width: '100', minWidth:
'50', maxWidth: '200' },
            { field: 'Progress', headerText: 'Progress', width: '150' }
        ]
    });
    gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column template in EJ2 JavaScript Gantt control

A column template is used to customize the column's look. The following code example explains how to define the custom template in Gantt using the [template](#) property.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
let ProjectResources: Object[] = [

```

```

    { resourceId: 1, resourceName: 'Martin Tamer' },
    { resourceId: 2, resourceName: 'Rose Fuller' },
    { resourceId: 3, resourceName: 'Margaret Buchanan' },
    { resourceId: 4, resourceName: 'Fuller King' },
    { resourceId: 5, resourceName: 'Davolio Fuller' },
    { resourceId: 6, resourceName: 'Van Jack' },
    { resourceId: 7, resourceName: 'Fuller Buchanan' },
    { resourceId: 8, resourceName: 'Jack Davolio' },
    { resourceId: 9, resourceName: 'Tamer Vinet' },
    { resourceId: 10, resourceName: 'Vinet Fuller' },
    { resourceId: 11, resourceName: 'Bergs Anton' },
    { resourceId: 12, resourceName: 'Construction Supervisor' }
  ];
let GanttData: Object[] = [
  {
    TaskID: 1,
    TaskName: 'Project initiation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      {TaskID: 2, TaskName: 'Identify site location', StartDate: new
Date('04/02/2019'), Duration: 4,Progress: 50, resources: [1]},
      {TaskID: 3, TaskName: 'Perform soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Predecessor: '2',Progress: 50, resources:
[2]},
      {TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Predecessor: '3', Progress: 50 ,resources:
[3]},
    ]
  },
  {
    TaskID: 5,
    TaskName: 'Project estimation',
    StartDate: new Date('04/02/2019'),
    EndDate: new Date('04/21/2019'),
    subtasks: [
      {TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'),Duration: 3, Predecessor: '4', Progress:
50, resources: [4]},
      {TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'),Duration: 3, Predecessor: '6', resources: [3],Progress:
50},
      {TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'),Duration: 3, Predecessor: '7',
    }
    ]
  }
];
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    resourceInfo: 'resources',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  }
});

```

```

    },
    rowHeight: 50,
    splitterSettings: {
        columnIndex: 3
    },
    height: '450px',
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
        { field: 'resources', headerText: 'Resources', width: '250' },
    template: '#columnTemplate' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName',
    },
    resources: ProjectResources
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script type="text/x-jsrender" id="columnTemplate">
    ${if(ganttProperties.resourceNames) }

    <div class="image">
      <div style="display:inline-
block;width:100%;position:relative;left:30px;top:-
14px">${gantttProperties.resourceNames}</div>
    </div>
    ${/if}
  </script>

```



```

<div id="container">
  <div id="Gantt"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column menu in EJ2 JavaScript Gantt control

The column menu has options to integrate features like sorting, filtering, and autofit. It will show a menu with the integrated feature when users click the Multiple icon of the column. To enable the column menu, you should set the [showColumnMenu](#) property to true.

The default items are displayed in the following table:

Item	Description
SortAscending	Sort the current column in ascending order.
SortDescending	Sort the current column in descending order.
AutoFit	Auto fit the current column.
AutoFitAll	Auto fit all columns.
Filter	Show the filter option as given in the <code>filterSettings.type</code> property.

INDEX.TS

```

import { Gantt, Filter, Sort, Resize, ColumnMenu } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter, Sort, Resize, ColumnMenu);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  showColumnMenu: true,
  allowFiltering: true,
  allowResizing: true,
  allowSorting: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  splitterSettings: {
    position: '100%'
  },
  height: '450px'
});

```

```
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

You can disable the column menu for a particular column by setting the `columns.showColumnMenu` to `false`.

Column menu events

During the resizing action, the gantt component triggers the below two events.

1. The [columnMenuOpen](#) event triggers before the column menu opens.
2. The [columnMenuClick](#) event triggers when the user clicks the column menu of the gantt.

INDEX.TS

```
import { Gantt, Filter, Sort, Resize, ColumnMenu } from '@syncfusion/ej2-
gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Filter, Sort, Resize, ColumnMenu);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
```

```

showColumnMenu: true,
allowFiltering: true,
allowResizing: true,
allowSorting: true,
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
},
splitterSettings: {
    position: '100%'
},
height: '450px',
columnMenuOpen: () => {
    alert('columnMenuOpen event is Triggered');
},
columnMenuClick: () => {
    alert('columnMenuClick event is Triggered');
}
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Custom Column Menu Item

Custom column menu items can be added by defining the [columnMenuItems](#). Actions for this customized items can be defined in the [columnMenuClick](#) event.

INDEX.TS

```
import { Gantt, Filter, Sort, Resize, ColumnMenu } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { MenuEventArgs } from '@syncfusion/ej2-navigations';
Gantt.Inject(Filter, Sort, Resize, ColumnMenu);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  showColumnMenu: true,
  allowFiltering: true,
  allowResizing: true,
  allowSorting: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  columnMenuItems: [{ text: 'Clear Sorting', id: 'ganttclearsorting' }],
  columnMenuClick: function(args: MenuEventArgs) {
    if(args.item.id === 'ganttclearsorting') {
      gantt.clearSorting();
    }
  },
  sortSettings: {
    columns: [{ direction: "Ascending", field: "TaskName" }]
  },
  splitterSettings: {
    position: '75%'
  },
  columns: [
    { field: 'TaskID', headerText: 'Task ID' },
    { field: 'TaskName', headerText: 'Task Name' },
    { field: 'StartDate', headerText: 'Start Date', width: '150' },
    { field: 'Duration', headerText: 'Duration', width: '100' },
    { field: 'Progress', headerText: 'Progress', width: '150' }
  ],
  height: '450px'
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize menu items for particular columns

Sometimes, you have a scenario that to hide an item from column menu for particular columns. In that case, you need to define the [columnMenuOpenEventArgs.hide](#) as true in the [columnMenuOpen](#) event.

The following sample, **Filter** item was hidden in column menu when opens for the **Task Name** column.

INDEX.TS

```

import { Gantt, Filter, Sort, Resize, ColumnMenu } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ColumnMenuOpenEventArgs, ColumnMenuItemModel } from '@syncfusion/ej2-grids';
Gantt.Inject(Filter, Sort, Resize, ColumnMenu);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    showColumnMenu: true,
    allowFiltering: true,
    allowResizing: true,
    allowSorting: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
    }
});

```

```

        child: 'subtasks',
    },
    splitterSettings: {
        position: '100%'
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        { field: 'TaskName', headerText: 'Task Name' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '100' },
        { field: 'Progress', headerText: 'Progress', width: '150' }
    ],
    height: '450px',
    columnMenuOpen: function (args: ColumnMenuOpenEventArgs) {
        for (let item of args.items) {
            if (item.text === 'Filter' && args.column.field === 'TaskName')
            {
                (item as ColumnMenuItemModel).hide = true;
            } else {
                (item as ColumnMenuItemModel).hide = false;
            }
        }
    },
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Responsive columns in EJ2 JavaScript Gantt control

You can toggle the column visibility based on media queries, which are defined in the [hideAtMedia](#). The [hideAtMedia](#) accepts valid [Media Queries](#).

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  splitterSettings: {
    position: '75%'
  },
  height: '450px',
  columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100' },
    { field: 'TaskName', headerText: 'Task Name', width: '200',
hideAtMedia: '(min-width: 700px)' },
    { field: 'StartDate', headerText: 'Start Date', width: '150' },
    { field: 'Duration', headerText: 'Duration', width: '100',
hideAtMedia: '(max-width: 500px)' },
    { field: 'Progress', headerText: 'Progress', width: '150' }
  ]
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change tree/expander column

The tree/expander column is a column in the Gantt control, that has icons to expand or collapse the parent records. You can define the tree column index in the Gantt control by using the [treeColumnIndex](#) property and the default value of this property is 0. The following code example shows how to use this property.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings: {
        columnIndex: 3
    },
    height: '450px',
    treeColumnIndex: 2
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or Hide columns dynamically

You can show or hide gantt columns dynamically using external buttons by invoking the [showColumn](#) or [hideColumn](#) method.

INDEX.TS

```

import { Gantt, Edit, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    splitterSettings: {
        position: '75%'
    },
    editSettings: {
        allowEditing: true
    },
    height: '450px',
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        { field: 'Progress', headerText: 'Progress' },
        { field: 'TaskName', headerText: 'Task Name' },
    ]
});

```

```

        { field: 'StartDate', headerText: 'Start Date' },
        { field: 'Duration', headerText: 'Duration' }
    ]
});
gantt.appendTo('#Gantt');
let show: Button = new Button();
show.appendTo('#show');
let hide: Button = new Button();
hide.appendTo('#hide');
document.getElementById('show').addEventListener('click', () => {
    gantt.showColumn(['TaskName', 'Duration']);
});
document.getElementById('hide').addEventListener('click', () => {
    gantt.hideColumn(['TaskName', 'Duration']);
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="show">Show</button>
        <button id="hide">Hide</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Check box columns in EJ2 JavaScript Gantt control

To render boolean values as checkbox in columns, you need to set [displayAsCheckBox](#) property as **true**.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
    verified: 'verified'
  },
  splitterSettings: {
    columnIndex: 5
  },
  height: '450px',
  columns: [
    { field: 'TaskID', headerText: 'Task ID' },
    { field: 'Progress', headerText: 'Progress' },
    { field: 'TaskName', headerText: 'Task Name' },
    { field: 'StartDate', headerText: 'Start Date' },
    { field: 'verified', headerText: 'Verified', displayAsCheckBox:
true, type: 'boolean' },
    { field: 'Duration', headerText: 'Duration' }
  ]
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Controlling Grid actions

You can enable or disable gantt action for a particular column by setting the [allowFiltering](#), [allowSorting](#), [allowReordering](#), and [allowEditing](#) properties.

INDEX.TS

```
import { Gantt, Filter, Sort, Resize, ColumnMenu, Reorder, Selection, Edit }
from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, Selection, Filter, Sort, Resize, Reorder);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowFiltering: true,
    allowSorting: true,
    allowReordering: true,
    splitterSettings: {
        position: '75%'
    },
    editSettings: {
        allowEditing: true
    },
    height: '450px',
    columns: [
        { field: 'TaskID', headerText: 'Task ID' },
        { field: 'Progress', headerText: 'Progress', allowReordering: false },
        { field: 'TaskName', headerText: 'Task Name', allowSorting: false },
        { field: 'StartDate', headerText: 'Start Date', allowEditing: false },
        { field: 'Duration', headerText: 'Duration', allowFiltering: false }
    ]
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Gantt</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column type

Column type can be specified using the `columns.type` property. It specifies the type of data the column binds.

If the `format` is defined for a column, the column uses `type` to select the appropriate format option [number](#) or [date](#).

Gantt column supports the following types:

- string
- number
- boolean
- date
- date time

If the `type` is not defined, it will be determined from the first record of the [dataSource](#).

In case if the first record of the [dataSource](#) is null/blank value for a column then it is necessary to define the `type` for that column.

Column spanning in EJ2 JavaScript Gantt control

The gantt has option to span the adjacent cells. You need to define the `colSpan` attribute to span cells in the [QueryCellInfo](#) event.

In the following demo, **Work 1** cells have been spanned.

INDEX.TS

```
import { Gantt, Edit, Selection, QueryCellInfoEventArgs } from
 '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  gridLines: 'Both',
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    work1: 'work1',
    work2: 'work2',
    progress: 'Progress',
    child: 'subtasks'
  },
  columns: [
    { field: 'TaskID', headerText: 'Task ID' },
    { field: 'TaskName', headerText: 'Task Name' },
    { field: 'work1', headerText: 'Work 1' },
    { field: 'work2', headerText: 'Work 2' },
    { field: 'StartDate', headerText: 'Start Date' },
    { field: 'Duration', headerText: 'Duration' },
    { field: 'Progress', headerText: 'Progress' }
  ],
  splitterSettings: {
    columnIndex: 5
  },
  queryCellInfo: function (args: QueryCellInfoEventArgs) {
    switch (args.data.TaskID) {
      case 1:
        if ((args.column.field == 'work1') && (args.data.taskData.work1
== 'support')) {
          args.colSpan = 2;
        }
        break;
      case 2:
        if ((args.column.field == 'work1') && (args.data.taskData.work1
== 'support')) {
          args.colSpan = 2;
        }
        break;
      case 3:
        if ((args.column.field == 'work1') && (args.data.taskData.work1
== 'support')) {
          args.colSpan = 2;
        }
        break;
      case 4:
        if ((args.column.field == 'work1') && (args.data.taskData.work1
== 'support')) {
```

```

        args.colSpan = 2;
    }
    break;
    case 5 :
        if ((args.column.field == 'work1') && (args.data.taskData.work1
== 'support')) {
            args.colSpan = 2;
        }
        break;
    case 7:
        if ((args.column.field == 'work1') && (args.data.taskData.work1
== 'support')) {
            args.colSpan = 2;
        }
        break;
    }
}
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

State persistence in EJ2 JavaScript Gantt control

State persistence refers to the Gantt's state maintained in the browser's [localStorage](#) even if the browser is refreshed or if you move to the next page within the browser. State persistence stores gantt's model object in the local storage when the [enablePersistence](#) is defined as true.

Get or set localStorage value

If the [enablePersistence](#) property is set to true, the gantt property value is saved in the `window.localStorage` for reference. You can get/set the localStorage value by using the `getItem/setItem` method in the `window.localStorage`.

```
`ts
//get the Gantt model.

let value: string = window.localStorage.getItem('ganttGantt'); //"ganttGantt" is component name +
component id.

let model: Object = JSON.parse(model);
,

`ts
//set the Gantt model.

window.localStorage.setItem('ganttGantt', JSON.stringify(model)); //"ganttGantt" is component name +
component id.
,
```

You can refer to our [JavaScript Gantt](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Gantt example](#) to know how to present and manipulate data.

Prevent columns from persisting

When the [enablePersistence](#) property is set to true, the Gantt properties such as [Filtering](#), [Sorting](#), and [Columns](#) will persist. You can use the `addOnPersist` method to prevent these Gantt properties from persisting.

The following example demonstrates how to prevent Gantt columns from persisting. In the [dataBound](#) event of the Gantt, you can override the `addOnPersist` method and remove the columns from the key list given for persistence.

Note: When the `enablePersistence` property is set to true, the Gantt features such as column template, column formatter, header text, and value accessor will not persist.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
```



```

        progress: 'Progress',
        child: 'subtasks',
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
120 },
        { field: 'TaskName', headerText: 'Task Name', width: 150},
        { field: 'StartDate', headerText: 'StartDate', width: 150 },
        { field: 'Duration', headerText: 'Duration', width: 150},
    ],
    enablePersistence: true,
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true,
        showDeleteConfirmDialog: true
    },
    dataBound: dataBound
});
gantt.appendTo('#Gantt');
function dataBound(args: any) {
    let cloned = this.addOnPersist;
    this.addOnPersist = function (key: any) {
        key = key.filter((item: string) => item !== "columns");
        return cloned.call(this, key);
    };
}
document.getElementById('add').onclick = () => {
    let obj = { field: "Progress", headerText: 'Progress', width: 100 };
    gantt.columns.push(obj as any); //you can add the columns by using the
Gantt columns method
    gantt.treeGrid.refreshColumns();
};
document.getElementById('remove').onclick = () => {
    gantt.columns.pop();
    gantt.treeGrid.refreshColumns();
};
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="add">Add columns</button>
        <button id="remove">Remove columns</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Persist the header template and header Text

By default, the Gantt properties such as column template, header text, header template, column formatter, and value accessor will not persist when [enablePersistence](#) is set to true. Because the column template and header text can be customized at the application level.

If you wish to restore all these column properties, then you can achieve it by cloning the gantt's columns property using JavaScript Object's assign method and storing this along with the persist data manually. While restoring the settings, this column object must be assigned to the gantt's columns property to restore the column settings as demonstrated in the following sample.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
120 },
        { field: 'TaskName', headerText: 'Task Name', width: 150,
headerTemplate: '#customertemplate' },
        { field: 'StartDate', headerText: 'StartDate', width: 150 },
        { field: 'Duration', headerText: 'Duration', width: 150 },
        { field: 'Progress', headerText: 'Progress', width: 150 },
    ],
    enablePersistence: true,

```

```

        editSettings: {
            allowAdding: true,
            allowEditing: true,
            allowDeleting: true,
            allowTaskbarEditing: true,
            showDeleteConfirmDialog: true
        }
    });
    gantt.appendTo('#Gantt');
    let savedProperties: any;
    document.getElementById('restore').onclick = () => {
        savedProperties = JSON.parse(gantt.getPersistData());
        var gridColumnState = Object.assign([], gantt.ganttColumns);
        savedProperties.columns.forEach((col: {
            field: any;
            headerText: any;
            template: any;
            headerTemplate: any;
        }) => {
            let headerText = gridColumnState.find((colColumnsState) =>
colColumnsState.field === col.field)['headerText'];
            let colTemplate = gridColumnState.find((colColumnsState) =>
colColumnsState.field === col.field)['template'];
            let headerTemplate = gridColumnState.find((colColumnsState) =>
colColumnsState.field === col.field)['headerTemplate'];
            col.headerText = 'Text Changed';
            col.template = colTemplate;
            col.headerTemplate = headerTemplate; //likewise you can restore
required column properties as per your wants.
        }
    );
    console.log(savedProperties);
    gantt.treeGrid.setProperties(savedProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <style>

    .e-column:before {
        content: '\e815';
    }
    .e-header:before {
        content: '\ea9a';
    }

```

```

    }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <script id="customertemplate" type="text/x-template">
        <span class="e-icons e-header" ></span>
        Task Name
    </script>

    <div id="container">
        <button id="restore">Restore</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tool bar in EJ2 JavaScript Gantt control

The Gantt control provides the toolbar support to handle Gantt actions. The [toolbar](#) property accepts the collection of built-in toolbar items and `ItemModel` objects for custom toolbar items.

Built-in toolbar items

Built-in toolbar items execute standard actions of the Gantt control, and these items can be added to toolbar by defining the [toolbar](#) as a collection of built-in items. It renders the button with icon and text.

The following table shows built-in toolbar items and its actions.

Built-in Toolbar Items	Actions
-----	-----
Add	Adds a new record.
Cancel	Cancels the edit state.
CollapseAll	Collapses all the rows.
Delete	Deletes the selected record.
Edit	Edits the selected record.
Indent	Indent the selected record to one level.
Outdent	Outdent the selected record to one level.
ExpandAll	Expands all the rows.

- | NextTimeSpan | Navigate the Gantt timeline to next time span. |
- | PrevTimeSpan | Navigate the Gantt timeline to previous time span. |
- | Search | Searches the records by the given key. |
- | Update | Updates the edited record. |

INDEX.TS

```
import { Gantt, Toolbar, Selection, Edit, Filter } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Selection, Edit, Filter);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  toolbar: ['Add', 'Cancel', 'CollapseAll', 'Delete', 'Edit', 'ExpandAll', 'NextTimeSpan', 'PrevTimeSpan', 'Search', 'Update', 'Indent', 'Outdent'],
  editSettings: {
    allowEditing: true,
    allowAdding: true,
    allowDeleting: true
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

    <div id="Gantt"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The [toolbar](#) has options to define both built-in and custom toolbar items.

Custom toolbar items

Custom toolbar items can be added to the toolbar by defining the [toolbar](#) property as a collection of `ItemModels`.

Actions for this customized toolbar items are defined in the [toolbarClick](#) event.

By default, the custom toolbar items are at left position. You can change the position by using the `align` property. In the following sample, the `Quick Filter` toolbar item is positioned at right.

INDEX.TS

```

import { Gantt, Toolbar, Filter } from '@syncfusion/ej2-gantt';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Filter);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  toolbarClick: (args: ClickEventArgs) => {
    if (args.item.id === 'toolbarfilter') {
      gantt.filterByColumn('TaskName', 'startswith', 'Identify');
    }
  },
  toolbar: [{ text: 'Quick Filter', tooltipText: 'Quick Filter', id:
'toolbarfilter', align: 'Right', prefixIcon: 'e-quickfilter' }],
  allowFiltering: true
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The [toolbar](#) has options to define both built-in and custom toolbar items.

* If a toolbar item does not match the built-in items, it will be treated as a custom toolbar item.

Built-in and custom items in toolbar

The Gantt control has an option to use both built-in and custom toolbar items at the same time.

In the following example, the **ExpandAll** and **CollapseAll** are built-in toolbar items and **Test** is the custom toolbar item.

INDEX.TS

```

import { Gantt, Toolbar, Filter } from '@syncfusion/ej2-gantt';
import { EmitType } from '@syncfusion/ej2-base';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Filter);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
  if (args.item.text === 'Test') {
    alert("Custom toolbar click...");
  }
};
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',

```

```

        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbarClick: clickHandler,
    toolbar: ['ExpandAll', 'CollapseAll', { text: 'Test', tooltipText:
'Test', id: 'Test' }]
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable/disable toolbar items

You can enable or disable the toolbar items by using the `enableItems` method.

INDEX.TS

```

import { Gantt, Toolbar, Filter } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Filter);
let gantt: Gantt = new Gantt({

```



```

dataSource: GanttData,
height: '450px',
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
},
toolbar: ['QuickFilter', 'ClearFilter'],
toolbarClick: (args: ClickEventArgs) => {
    if (args.item.text === 'QuickFilter') {
        gantt.filterByColumn('TaskName', 'startswith', 'Identify');
    }
    if (args.item.text === 'ClearFilter') {
        gantt.clearFiltering();
    }
},
allowFiltering: true
});
gantt.appendTo('#Gantt');
let enable: Button = new Button({}, '#enable');
let disable: Button = new Button({}, '#disable');
enable.element.onclick = () => {
    gantt.toolbarModule.enableItems([gantt.element.id + '_QuickFilter',
    gantt.element.id + '_ClearFilter'], true); // enable toolbar items.
};
disable.element.onclick = () => {
    gantt.toolbarModule.enableItems([gantt.element.id + '_QuickFilter',
    gantt.element.id + '_ClearFilter'], false); // disable toolbar items.
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <button id="enable" class="e-flat">Enable</button>
        <button id="disable" class="e-flat">Disable</button>
        <div id="Gantt"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add input elements to toolbar

In the Gantt toolbar, you can add EJ2 editor elements like numeric text box, drop-down list, and date picker controls. The following code snippets demonstrates how to add EJ2 editors to the Gantt toolbar.

INDEX.TS

```

import { Gantt, Toolbar } from '@syncfusion/ej2-gantt';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    toolbar: [{ type: 'Input', template: new NumericTextBox({ format: 'c2',
value:1, width:150 }) }],
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Context menu in EJ2 JavaScript Gantt control

The Gantt control allows you to perform quick actions by using context menu. When right-clicking the context menu, the context menu options are shown. To enable this feature, set the [enableContextMenu](#) to true. The default context menu options are enabled using the [editSettings](#) property. The context menu options can be customized using the [contextMenuItems](#) property.

To use the context menu, inject the [ContextMenu](#) module into the Gantt control.

The default items are listed in the following table.

Items | Description

AutoFit | Auto-fits the current column.

AutoFitAll | Auto-fits all columns.

SortAscending | Sorts the current column in ascending order.

SortDescending | Sorts the current column in descending order.

TaskInformation | Edits the current task.

Add | Adds a new row to the Gantt.

Indent | Indent the selected record to one level.

Outdent | Outdent the selected record to one level.

DeleteTask | Deletes the current task.

Save | Saves the edited task.

Cancel | Cancels the edited task.

DeleteDependency | Deletes the current dependency task link.

Convert | Converts current task to milestone or vice-versa.

INDEX.TS

```
import { Gantt, Resize, Sort, ContextMenu, Edit, Selection } from
'@syncfusion/ej2-gantt';
import { GanttData } from './datasource.ts';
Gantt.Inject(Resize, Sort, Edit, ContextMenu, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    dependency: 'Predecessor',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  enableContextMenu: true,
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true
  }
});
gantt.appendTo('#ContextMenu');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="ContextMenu"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Custom context menu items

The custom context menu items can be added by defining the [contextMenuItems](#) as a collection of [contextMenuItemModel](#). Actions for the customized items can be defined in the [contextMenuClick](#) event and the visibility of customized items can be defined in the [contextMenuOpen](#) event.

To create custom context menu items for header area, define the target property as `.e-gridheader`.

The following sample shows context menu item for parent rows to expand or collapse child rows in the content area and a context menu item to hide columns in the header area.

INDEX.TS

```
import { Gantt, Selection, Toolbar, Edit, Resize, Sort, ContextMenu,
ContextMenuClickEventArgs, ContextMenuOpenEventArgs, ContextMenuItem } from
'@syncfusion/ej2-gantt';
import { ContextMenuItemModel } from '@syncfusion/ej2-grids';
import { GanttData } from './datasource.ts';
Gantt.Inject(Selection, Toolbar, Edit, Resize, Sort, ContextMenu, );
let contextMenuItems: (string | ContextMenuItemModel)[] = ['AutoFitAll',
'AutoFit', 'TaskInformation', 'DeleteTask', 'Save', 'Cancel',
'SortAscending', 'SortDescending', 'Add', 'DeleteDependency',
'Convert',
{ text: 'Collapse the Row', target: '.e-content', id: 'collapserow'
} as ContextMenuItemModel,
{ text: 'Expand the Row', target: '.e-content', id: 'expandrow' } as
ContextMenuItemModel,
{ text: 'Hide Column', target: '.e-gridheader', id: 'hidecols' } as
ContextMenuItemModel,
];
let gantt: Gantt = new Gantt({
dataSource: GanttData,
taskFields: {
id: 'TaskID',
name: 'TaskName',
startDate: 'StartDate',
duration: 'Duration',
progress: 'Progress',
dependency: 'Predecessor',
child: 'subtasks'
},
editSettings: {
allowAdding: true,
allowEditing: true,
allowDeleting: true
},
enableContextMenu: true,
allowSorting: true,
allowResizing: true,
contextMenuItems: contextMenuItems as ContextMenuItem[],
contextMenuClick: (args?: ContextMenuClickEventArgs) => {
let record = args.rowData;
if (args.item.id === 'collapserow') {
gantt.collapseByID(Number(record.ganttProperties.taskId));
```

```

    }
    if (args.item.id === 'expandrow') {
        gantt.expandById(Number(record.ganttProperties.taskId));
    }
    if (args.item.id === 'hidecols') {
        gantt.hideColumn(args.column.headerText);
    }
},
contextMenuOpen: (args?: ContextMenuOpenEventArgs) => {
    let record = args.rowData;
    if (args.type !== 'Header') {
        if (!record.hasChildRecords) {
            args.hideItems.push('Collapse the Row');
            args.hideItems.push('Expand the Row');
        } else {
            if (record.expanded) {
                args.hideItems.push("Expand the Row");
            } else {
                args.hideItems.push("Collapse the Row");
            }
        }
    }
}
});
gantt.appendTo('#CustomContextMenu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="CustomContextMenu"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can show an specific item in context menu for header/content area in the Gantt control by defining the **target** property.

Excel Export

Excel export in EJ2 JavaScript Gantt control

Gantt supports client-side exporting, which allows you to export its data to the Excel and CSV formats. Use the [excelExport](#) and [csvExport](#) methods for exporting. To enable Excel export in the Gantt, set the [allowExcelExport](#) to true.

To export data to Excel and CSV, inject the **ExcelExport** module in Gantt.

INDEX.TS

```

import { Gantt, Toolbar, ExcelExport, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, ExcelExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_excelexport') {
        gantt.excelExport();
    }
    else if (args.item.id === 'GanttExport_csvexport') {
        gantt.csvExport();
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowExcelExport: true,
    toolbar: ['ExcelExport', 'CsvExport'],
    toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom data source in EJ2 JavaScript Gantt control

The excel export provides an option to define datasource dynamically before exporting. To export data dynamically, define the `dataSource` in `exportProperties`.

INDEX.TS

```

import { Gantt, Toolbar, ExcelExport, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, ExcelExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_excelexport') {
        let excelExportProperties: ExcelExportProperties = {
            dataSource: [GanttData[1]]
        };
        gantt.excelExport(excelExportProperties);
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowExcelExport: true,
    toolbar: ['ExcelExport'],

```



```

        toolbarClick: clickHandler
    });
    gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple gantt exporting in EJ2 JavaScript Gantt control

In Gantt, the Excel export provides support to export multiple Gantt data in new sheet or same sheet.

Same sheet

The Excel export provides support to export multiple Gantt data in the same sheet. To export in same sheet, define `multipleExport.type` as `AppendToSheet` in `ExcelExportProperties`. You can also provide blank rows between exported multiple Gantt data. These blank rows count can be defined using `multipleExport.blankRows`.

INDEX.TS

```

import { Gantt, Toolbar, ExcelExport, Selection } from '@syncfusion/ej2-
gantt';
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, ExcelExport, Selection);
let firstGantt: Gantt = new Gantt({

```

```

        dataSource: [GanttData[0]],
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        },
        treeColumnIndex: 1,
        allowExcelExport: true,
        projectStartDate: new Date('03/31/2019'),
        projectEndDate: new Date('04/14/2019'),
        height: '280px',
        toolbar: ['ExcelExport']
    });
    firstGantt.appendTo('#GanttExport1');
    let secondGantt: Gantt = new Gantt({
        dataSource: [GanttData[1]],
        taskFields: {
            id: 'TaskID',
            name: 'TaskName',
            startDate: 'StartDate',
            duration: 'Duration',
            progress: 'Progress',
            child: 'subtasks'
        },
        treeColumnIndex: 1,
        allowExcelExport: true,
        height: '250px'
    });
    secondGantt.appendTo('#GanttExport2');
    firstGantt.toolbarClick = (args: Object) => {
        if (args.item.id === 'GanttExport1_excelexport') {
            let appendExcelExportProperties: ExcelExportProperties = {
                multipleExport: { type: 'AppendToSheet', blankRows: 2 }
            };
            let firstGanttExport: Promise<any> =
            firstGantt.excelExport(appendExcelExportProperties, true);
            firstGanttExport.then((fData: any) => {
                secondGantt.excelExport(appendExcelExportProperties, false,
                fData);
            });
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport1"></div>
        <div id="GanttExport2" style="margin-top:10px"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, `multipleExport.blankRows` value is 5.

New sheet

The Excel exporting provides support to export multiple Gantt in new sheet. To export in new sheet, define `multipleExport.type` as `NewSheet` in `ExcelExportProperties`.

INDEX.TS

```

import { Gantt, Toolbar, ExcelExport, Selection } from '@syncfusion/ej2-gantt';
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, ExcelExport, Selection);
let firstGantt: Gantt = new Gantt({
    dataSource: [GanttData[0]],
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    treeColumnIndex: 1,
    height: '280px',
    projectStartDate: new Date('03/31/2019'),
    projectEndDate: new Date('04/14/2019'),
    allowExcelExport: true,
    toolbar: ['ExcelExport']
});
firstGantt.appendTo('#GanttExport1');

```

```

let secondGantt: Gantt = new Gantt({
    dataSource: [GanttData[1]],
    height: '250px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    treeColumnIndex: 1,
    allowExcelExport: true
});
secondGantt.appendTo('#GanttExport2');
firstGantt.toolbarClick = (args: Object) => {
    if (args.item.id === 'GanttExport1_excelexport') {
        let appendExcelExportProperties: ExcelExportProperties = {
            multipleExport: { type: 'NewSheet' }
        };
        let firstGanttExport: Promise<any> =
firstGantt.excelExport(appendExcelExportProperties, true);
        firstGanttExport.then((fData: any) => {
            secondGantt.excelExport(appendExcelExportProperties, false,
fData);
        });
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport1"></div>
        <div id="GanttExport2" style="margin-top:10px"></div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the Excel export

Gantt Excel export allows the users to customize the exported document based on requirement.

Export hidden columns

In Gantt, the Excel export provides an option to export hidden columns by defining `includeHiddenColumn` as `true`.

INDEX.TS

```

import { Gantt, Toolbar, ExcelExport, Selection } from '@syncfusion/ej2-gantt';
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, ExcelExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_excelexport') {
        let excelExportProperties: ExcelExportProperties = {
            includeHiddenColumn: true
        };
        gantt.excelExport(excelExportProperties);
    }
    else if (args.item.id === 'GanttExport_csvexport') {
        let excelExportProperties: ExcelExportProperties = {
            includeHiddenColumn: true
        };
        gantt.csvExport(excelExportProperties);
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    treeColumnIndex: 1,
    allowExcelExport: true,
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width: '100' },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
        { field: 'StartDate', headerText: 'Start Date', width: '150', visible: false },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ]
});

```

```

    ],
    toolbar: ['ExcelExport', 'CsvExport'],
    toolbarClick: clickHandler
  });
  gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide columns on exported Excel

In Gantt, while exporting, you can show a hidden column or hide a visible column using the [toolbarClick](#) and [excelExportComplete](#) events.

In the `toolbarClick` event, using the `args.item.id` property, you can show or hide columns by setting the `column.visible` property to `true` or `false` respectively.

Similarly, in the `excelExportComplete` event, you can revert the columns visibility back to the previous state.

INDEX.TS

```

import { Gantt, Toolbar, ExcelExport, Selection } from '@syncfusion/ej2-
gantt';
import { GanttData } from 'datasource.ts';

```

```

Gantt.Inject(Toolbar, ExcelExport, Selection);
var clickHandler = function(args){
    let columns: Column[] = gantt.treeGrid.grid.columns;
    if (args.item.id === 'GanttExport_excelexport') {
        columns[0].visible = true;
        columns[3].visible = false;
        gantt.excelExport();
    }
    else if (args.item.id === 'GanttExport_csvexport') {
        columns[0].visible = true;
        columns[3].visible = false;
        gantt.csvExport();
    }
};
var exportComplete = function(args){
    let columns: Column[] = gantt.treeGrid.grid.columns;
    columns[0].visible = false;
    columns[3].visible = true;
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    treeColumnIndex: 1,
    allowExcelExport: true,
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100',visible:false },
        { field: 'TaskName', headerText: 'Task Name', width: '150' },
        { field: 'StartDate', headerText: 'Start Date', width: '150' },
        { field: 'Duration', headerText: 'Duration', width: '150' },
        { field: 'Progress', headerText: 'Progress', width: '150' },
    ],
    toolbar: ['ExcelExport', 'CsvExport'],
    toolbarClick: clickHandler,
    excelExportComplete: exportComplete
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell formatting during export

In Gantt, you can customize the TreeGrid cells in the exported document using the [excelQueryCellInfo](#) event. In this event, you can format the TreeGrid cells of exported Excel and CSV documents based on the required condition.

In the following sample, the background color has been customized for **TaskID** column in the exported Excel using the **args.style** and **backColor** properties.

INDEX.TS

```

import { Gantt, Toolbar, ExcelExport, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, ExcelExport, Selection);
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    treeColumnIndex: 1,
    allowExcelExport: true,
    labelSettings: {
        taskLabel: '${Progress}%'
    },
    toolbar: ['ExcelExport'],

```



```

        splitterSettings: {
            columnIndex: 3
        },
        columns: [
            { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left', width:
'100', visible: false },
            { field: 'TaskName', headerText: 'Task Name', width: '150' },
            { field: 'Progress', headerText: 'Progress', width: '150' },
            { field: 'StartDate', headerText: 'Start Date', width: '150' },
            { field: 'Duration', headerText: 'Duration', width: '150' }
        ],
        excelQueryCellInfo: function (args: any) {
            if (args.column.field == 'Progress') {
                if (args.value > 80) {
                    args.style = { backgroundColor: '#A569BD' };
                }
                else if (args.value < 20) {
                    args.style = { backgroundColor: '#F08080' };
                }
            }
        },
        queryTaskbarInfo: function (args: any) {
            if (args.data.Progress > 80) {
                args.progressBarBgColor = "#6C3483";
                args.taskbarBgColor = args.taskbarBorderColor = "#A569BD";
            } else if (args.data.Progress < 20) {
                args.progressBarBgColor = "#CD5C5C";
                args.taskbarBgColor = args.taskbarBorderColor = "#F08080";
            }
        },
        queryCellInfo: function (args: any) {
            if (args.column.field == 'Progress') {
                if (args.data.Progress > 80) {
                    args.cell.style.backgroundColor = '#A569BD';
                }
                else if (args.data.Progress < 20) {
                    args.cell.style.backgroundColor = '#F08080';
                }
            }
        }
    });
    gantt.appendTo('#GanttExport');
    gantt.toolbarClick = (args: Object) => {
        if (args.item.id === 'GanttExport_excelexport') {
            gantt.excelExport();
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
      <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Theme

The Excel export also provides an option to include custom theme for exported Excel document.

To apply theme in exported Excel, define the **theme** in **ExcelExportProperties**.

INDEX.TS

```

import { Gantt, Toolbar, ExcelExport, Selection } from '@syncfusion/ej2-gantt';
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, ExcelExport, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  treeColumnIndex: 1,
  allowExcelExport: true,
  toolbar: ['ExcelExport']
});
gantt.appendTo('#GanttExport');
gantt.toolbarClick = (args: Object) => {
  if (args['item'].id === 'GanttExport_excelexport') {

```

```

    let excelExportProperties: ExcelExportProperties = {
        theme:
            {
                header: { fontName: 'Segoe UI', fontColor: '#666666' },
                record: { fontName: 'Segoe UI', fontColor: '#666666' }
            }
    };
    gantt.excelExport(excelExportProperties);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, material theme is applied to the exported Excel document.

Add header and footer

The Excel export also allows users to include header and footer contents to the exported Excel document.

INDEX.TS

```

import { Gantt, Toolbar, ExcelExport, Selection } from '@syncfusion/ej2-
gantt';
import { ExcelExportProperties } from '@syncfusion/ej2-grids'

```

```

import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, ExcelExport, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  treeColumnIndex: 1,
  allowExcelExport: true,
  toolbar: ['ExcelExport']
});
gantt.appendTo('#GanttExport');
gantt.toolbarClick = (args: Object) => {
  if (args['item'].id === 'GanttExport_excelexport') {
    let excelExportProperties: ExcelExportProperties = {
      header: {
        headerRows: 7,
        rows: [
          { cells: [{ colSpan: 4, value: "Northwind Traders",
style: { fontColor: '#C67878', fontSize: 20, hAlign: 'Center', bold: true, }
}} ],
          { cells: [{ colSpan: 4, value: "2501 Aerial Center
Parkway", style: { fontColor: '#C67878', fontSize: 15, hAlign: 'Center',
bold: true, } } ] },
          { cells: [{ colSpan: 4, value: "Suite 200 Morrisville,
NC 27560 USA", style: { fontColor: '#C67878', fontSize: 15, hAlign:
'Center', bold: true, } } ] },
          { cells: [{ colSpan: 4, value: "Tel +1 888.936.8638 Fax
+1 919.573.0306", style: { fontColor: '#C67878', fontSize: 15, hAlign:
'Center', bold: true, } } ] },
          { cells: [{ colSpan: 4, hyperlink: { target:
'https://www.northwind.com/', displayText: 'www.northwind.com' }, style: {
hAlign: 'Center' } } ] },
          { cells: [{ colSpan: 4, hyperlink: { target:
'mailto:support@northwind.com' }, style: { hAlign: 'Center' } } ] },
        ]
      },
      footer: {
        footerRows: 4,
        rows: [
          { cells: [{ colSpan: 4, value: "Thank you for your
business!", style: { hAlign: 'Center', bold: true } } ] },
          { cells: [{ colSpan: 4, value: "!Visit Again!", style: {
hAlign: 'Center', bold: true } } ] }
        ]
      },
    };
    gantt.excelExport(excelExportProperties);
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

File name for exported document

You can set the required file name for the exported document by defining the `fileName` property in `ExcelExportProperties`.

INDEX.TS

```

import { Gantt, Toolbar, ExcelExport, Selection } from '@syncfusion/ej2-
gantt';
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, ExcelExport, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',

```

```

        child: 'subtasks'
      },
      treeColumnIndex: 1,
      allowExcelExport: true,
      toolbar: ['ExcelExport', 'CsvExport']
    });
    gantt.appendTo('#GanttExport');
    gantt.toolbarClick = (args: Object) => {
      if (args['item'].id === 'GanttExport_excelexport') {
        let excelExportProperties: ExcelExportProperties = {
          fileName: "Gantt.xlsx"
        };
        gantt.excelExport(excelExportProperties);
      }
      else if (args['item'].id === 'GanttExport_csvexport') {
        let excelExportProperties: ExcelExportProperties = {
          fileName: "Gantt.csv"
        };
        gantt.csvExport(excelExportProperties);
      }
    }
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Pdf Export

Pdf export in EJ2 JavaScript Gantt control

PDF Export

PDF export allows exporting Gantt data to PDF document. You need to use the [pdfExport](#) method for exporting. To enable PDF export in the Gantt, set the [allowPdfExport](#) to true.

To export data to PDF document, inject the PdfExport module in Gantt.

INDEX.TS

```
import { Gantt, Toolbar, PdfExport, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        gantt.pdfExport();
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Indicators in PDF exporting

The PDF export functionality allows users to export Gantt charts enriched with dynamic indicators and accompanying images.

These indicators, represented by images, can be effortlessly defined using the [base64](#) encoding value in the data object of datasource. This data object field should be mapped to indicator property of [task fields](#).

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection } from '@syncfusion/ej2-
gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        gantt.pdfExport();
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks',
        indicators: 'Indicators'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting Gantt data as a blob object

In Gantt, you can export the Gantt chart data as a blob object, which allows you to preview or modify the data before exporting it.

To export the Gantt chart data as a blob object, follow these steps:

step 1: pdfExport fourth argument set as **true**.

step 2: Then , pdfExpComplete return as blob object.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection, ExcelExport,
PdfExportProperties, ExcelExportCompleteArgs, ExcelExportCompleteArgs } from
'@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection, ExcelExport);
/**
 * Exporting Blob data

```

```

*/
let excelExpComplete: EmitType<ExcelExportCompleteArgs> = (args:
ExcelExportCompleteArgs) => {
    //This event will be triggered when excel exporting.
    args.promise.then((e: { blobData: Blob }) => {
        //In this `then` function, we can get blob data through the
arguments after promise resolved.
        exportBlob(e.blobData);
    });
};

let pdfExpComplete: EmitType<ExcelExportCompleteArgs> = (args:
PdfExportCompleteArgs) => {
    //This event will be triggered when pdf exporting.
    args.promise.then((e: { blobData: Blob }) => {
        //In this `then` function, we can get blob data through the
arguments after promise resolved.
        exportBlob(e.blobData);
    });
};

let exportBlob: Function = (blob: Blob) => {
    let a: HTMLAnchorElement = document.createElement('a');
    document.body.appendChild(a);
    a.style.display = 'none';
    let url: string = window.URL.createObjectURL(blob);
    a.href = url;
    a.download = 'Export';
    a.click();
    window.URL.revokeObjectURL(url);
    document.body.removeChild(a);
}

let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        gantt.pdfExport(null, null, null, true);
    }
    if (args.item.id === 'GanttExport_excelexport') {
        gantt.excelExport(null, null, null, true);
    }
};

let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName' },
        { field: 'StartDate' },
        { field: 'Duration' },
        { field: 'Progress' }
    ],
    allowPdfExport: true,

```

```

        allowExcelExport: true,
        excelExportComplete: excelExpComplete,
        pdfExportComplete: pdfExpComplete,
        toolbar: ['PdfExport', 'ExcelExport'],
        toolbarClick: clickHandler
    });
    gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
  <script src="http://cdn.syncfusion.com/ej2/20.4.48/dist/ej2.min.js"
type="text/javascript"></script>

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Single page exporting in gantt

In Gantt, we have provided support to export the Gantt component where each rows are auto-fit to the PDF document page width by setting [isFitToWidth](#) as true in `fitToWidthSettings` of `PdfExportProperties`.

Also, we can customize the chart width and grid width in exported file using [chartWidth](#) and [gridWidth](#) by defining it as percentage in string.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection, PdfExportProperties } from
 '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        let exportProperties: PdfExportProperties = {
            fitToWidthSettings: {
                isFitToWidth: true,
            }
        };
        gantt.pdfExport(exportProperties);
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    dateFormat: 'MMM dd, y',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks',
    },
    columns: [
        { field: 'TaskID', width: 80 },
        { field: 'TaskName', width: 250 },
        { field: 'StartDate' },
        { field: 'EndDate' },
        { field: 'Duration' },
        { field: 'Predecessor' },
        { field: 'resources' },
        { field: 'Progress' }
    ],
    splitterSettings: {
        columnIndex: 2
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler,
    allowSelection: true,
    gridLines: 'Both',
    height: '450px',
    treeColumnIndex: 1,
    resourceFields: {
        id: 'resourceId',
        name: 'resourceName'
    },
    highlightWeekends: true,
    timelineSettings: {
        topTier: {
            unit: 'Week',

```

```

        format: 'MMM dd, y',
    },
    bottomTier: {
        unit: 'Day',
    },
    },
    labelSettings: {
        leftLabel: 'TaskName',
        rightLabel: 'resources'
    },
    projectStartDate: new Date('03/25/2019'),
    projectEndDate: new Date('07/28/2019')
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting with column template

The PDF export functionality allows to export Grid columns that include images, hyperlinks, and custom text to an PDF document using [pdfQueryCellInfo](#) event.

In the following sample, the hyperlinks and images are exported to PDF using [hyperlink](#) and [image](#) properties in the [pdfQueryCellInfo](#) event.

Note: PDF Export supports base64 string to export the images.

INDEX.TS

```
import { Gantt, Toolbar, PdfExport, Selection,
PdfExportProperties, PdfQueryCellInfoEventArgs } from '@syncfusion/ej2-
gantt';
import { GanttData, editingResources } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        let exportProperties: PdfExportProperties = {
            enableFooter: false
        };
        gantt.pdfExport(exportProperties);
    }
};
function pdfQueryCellInfo(args: PdfQueryCellInfoEventArgs): any {
    if (args.column.headerText === 'Resources') {
        {
            args.image = { height: 40, width: 40, base64: (args as
any).data.taskData.resourcesImage };
        }
    }
    if (args.column.headerText === 'Email ID') {
        args.hyperLink = {
            target: 'mailto:' + (args as any).data.taskData.EmailId,
            displayText: (args as any).data.taskData.EmailId
        };
    }
}
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        endDate: 'EndDate',
        duration: 'Duration',
        progress: 'Progress',
        resourceInfo: 'resources',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Left' },
        { field: 'TaskName', headerText: 'Task Name', width: '250' },
        { field: 'resources', headerText: 'Resources', width: '250',
template: '#columnTemplate' },
        { field: 'EmailId', headerText: 'Email ID', template: '#template2',
width: '180' },
    ],
    pdfQueryCellInfo: pdfQueryCellInfo,
    allowPdfExport: true,
```

```

        toolbar: ['PdfExport'],
        toolbarClick: clickHandler,
        resources: editingResources,
        resourceFields: {
            id: 'resourceId',
            name: 'resourceName'
        },
        projectStartDate: new Date('03/24/2019'),
        projectEndDate: new Date('07/06/2019')
    });
    gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css" rel="stylesheet"
type="text/css">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script type="text/x-jsrender" id="columnTemplate">
        ${if(ganttProperties.resourceNames)}
        <div class="image">
            
        </div>
        ${/if}
    </script>
    <script id="template2" type="text/x-template">
        ${if(taskData.EmailId)}
        <div class="link">
            <a href="mailto:${taskData.EmailId}">${taskData.EmailId}</a></div>
        </div>
        ${/if}
    </script>
    <div id="container">
        <div id="GanttExport"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Multiple gantt exporting in EJ2 JavaScript Gantt control

PDF export provides an option for exporting multiple Gantt to same file. In this exported document, each Gantt will be exported to a new page of the document in same file.

INDEX.TS

```
import { Gantt, Toolbar, PdfExport, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, PdfExport, Selection);
let firstGantt: Gantt = new Gantt({
  dataSource: [GanttData[0]],
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  treeColumnIndex: 1,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  projectStartDate: new Date('03/31/2019'),
  projectEndDate: new Date('04/14/2019'),
  height: '280px',
});
firstGantt.appendTo('#GanttExport1');
let secondGantt: Gantt = new Gantt({
  dataSource: [GanttData[1]],
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  treeColumnIndex: 1,
  toolbar: ['PdfExport'],
  height: '250px'
});
secondGantt.appendTo('#GanttExport2');
firstGantt.toolbarClick = (args: Object) => {
  if (args.item.id === 'GanttExport1_pdfexport') {
    let firstGanttPdfExport: Promise<Object> = gantt.pdfExport({},
true);
    firstGanttPdfExport.then(function(pdfData) {
      secondGantt.pdfExport({}, false, pdfData);
    });
  }
}
```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport1"></div>
    <div id="GanttExport2" style="margin-top:10px"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

To customize PDF export

PDF export provides an option to customize the mapping of Gantt to exported PDF document.

File name for exported document

You can assign a file name for the exported document by defining the `fileName` property in `pdfExportProperties`.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection, PdfExportProperties } from
'@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
  if (args.item.id === 'GanttExport_pdfexport') {
    let exportProperties: PdfExportProperties = {
      fileName: "new.pdf"
    };
    gantt.pdfExport(exportProperties);
  }
}

```

```

});
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

How to change page orientation

Page orientation can be changed to **Portrait** (Default Landscape) for the exported document using the property `pdfExportProperties.pageOrientation`.

INDEX.TS

```
import { Gantt, Toolbar, PdfExport, Selection, PdfExportProperties } from
'@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        let exportProperties: PdfExportProperties = {
            pageOrientation: 'Portrait'
        };
        gantt.pdfExport(exportProperties);
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
```

```
        </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

How to change page size

Page size can be customized for the exported document using the `pdfExportProperties.pageSize`.

The supported page sizes are:

- Letter
- Note
- Legal
- A0
- A1
- A2
- A3
- A5
- A6
- A7
- A8
- A9
- B0
- B1
- B2
- B3
- B4
- B5
- ArchA
- ArchB
- ArchC
- ArchD
- ArchE
- FlsA
- HalfLetter
- Letter11x17
- Ledger

INDEX.TS

```
import { Gantt, Toolbar, PdfExport, Selection, PdfExportProperties } from
'@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
```

```

import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        let exportProperties: PdfExportProperties = {
            pageSize: 'A0'
        };
        gantt.pdfExport(exportProperties);
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {

```

```

        ele.style.visibility = "visible";
    }
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export current view data

PDF export provides an option to export the current view data into PDF. To export current view data alone, define the `exportType` to `CurrentViewData`.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection, Filter, PdfExportProperties }
from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection, Filter);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        let exportProperties: PdfExportProperties = {
            exportType: 'CurrentViewData'
        };
        gantt.pdfExport(exportProperties);
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    allowFiltering: true,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable footer

By default, we render the default footer for a PDF file, this can be enabled or disabled by using the `enableFooter` property.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection, PdfExportProperties } from
'@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        let exportProperties: PdfExportProperties = {
            enableFooter: false
        };
        gantt.pdfExport(exportProperties);
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],

```

```

        toolbarClick: clickHandler
    });
    gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Export hidden columns

PDF export provides an option to export hidden columns of Gantt by defining the `includeHiddenColumn` to `true`.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection, PdfExportProperties } from
'@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
  if (args.item.id === 'GanttExport_pdfexport') {
    let exportProperties: PdfExportProperties = {
      includeHiddenColumn: true
    };
  };
};

```



```

        gantt.pdfExport (exportProperties);
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', visible: false },
        { field: 'StartDate' },
        { field: 'Duration' },
        { field: 'Progress' }
    ],
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export predecessor lines

By using `showPredecessorLines`, you can hide or show predecessor lines in the exported PDF document.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection, PdfExportProperties } from
 '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
 navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
  if (args.item.id === 'GanttExport_pdfexport') {
    let exportProperties: PdfExportProperties = {
      showPredecessorLines: true
    };
    gantt.pdfExport();
  }
};
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  columns: [
    { field: 'TaskID' },
    { field: 'TaskName', visible: false },
    { field: 'StartDate' },
    { field: 'Duration' },
    { field: 'Progress' }
  ],
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Gantt Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide columns on exported PDF

You can show a hidden column or hide a visible column while exporting the Gantt using the [toolbarClick](#) and [beforePdfExport](#) events.

You can show or hide columns by setting the `column.visible` property to `true` or `false` respectively.

In the following example, there is a hidden column `Duration` in the Gantt. While exporting, we have changed `Duration` to visible column and `StartDate` to hidden column.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection } from '@syncfusion/ej2-gantt';
import { EJ2Instance } from '@syncfusion/ej2-navigations';
import { ClickEventArgs } from '@syncfusion/ej2-navigations/src/toolbar/toolbar';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        let obj: any =
        (<EJ2Instance>document.getElementById('GanttExport')).ej2_instances[0]
        obj.treeGrid.columns[2].visible = false;
        gantt.pdfExport();
    }
};
let beforePdfExport: EmitType<Object> = (args: Object) => {

```

```

    let obj: any =
    (<EJ2Instance>document.getElementById('GanttExport')).ej2_instances[0]
    obj.treeGrid.columns[3].visible = true;
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName' },
        { field: 'StartDate' },
        { field: 'Duration', visible: false },
        { field: 'Progress' }
    ],
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler,
    beforePdfExport: beforePdfExport
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
    </div>

<script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Conditional cell formatting

TreeGrid cells in the exported PDF can be customized or formatted using the [pdfQueryCellInfo](#) event. In this event, you can format the treegrid cells of exported PDF document based on the column cell value.

In the following sample, the background color is set for **Progress** column in the exported document by using the `args.style` and `backgroundColor` properties.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection, PdfQueryTimelineCellInfoEventArgs } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        gantt.pdfExport();
    }
};
let pdfQueryCellInfo: EmitType<PdfQueryCellInfoEventArgs> = (args: PdfQueryCellInfoEventArgs) => {
    if (args.column.field === 'Progress') {
        if (args.value < 50) {
            args.style = { backgroundColor: '#F08080' };
        } else {
            args.style = { backgroundColor: '#A569BD' };
        }
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', visible: false },
        { field: 'StartDate' },
        { field: 'Duration' },
        { field: 'Progress' }
    ],

```

```

    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler,
    pdfQueryCellInfo: pdfQueryCellInfo
  });
  gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Timeline cell formatting

Timeline cells in the exported PDF document can be customized or formatted using the [pdfQueryTimelineCellInfo](#) event.

In the following sample, the header background color is set for timeline cells in the exported document by using the `args.headerBackgroundColor` property.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection,
PdfQueryTimelineCellInfoEventArgs } from '@syncfusion/ej2-gantt';
import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
import { PdfColor } from '@syncfusion/ej2-pdf-export';

```

```

import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        gantt.pdfExport();
    }
};
let pdfQueryTimelineCellInfo: EmitType<PdfQueryTimelineCellInfoEventArgs> =
(args: PdfQueryTimelineCellInfoEventArgs) => {
    args.timelineCell.backgroundColor= new PdfColor(240, 248, 255);
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', visible: false },
        { field: 'StartDate' },
        { field: 'Duration' },
        { field: 'Progress' }
    ],
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler,
    pdfQueryTimelineCellInfo: pdfQueryTimelineCellInfo
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

        <div id="container">
            <div id="GanttExport"></div>
        </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Taskbar formatting

Taskbars in the exported PDF document can be customized or formatted using the [pdfQueryTaskbarInfo](#) event.

In the following sample, the taskbar background color is customized in the chart side of the exported document by using the `args.taskbar` property.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection } from '@syncfusion/ej2-gantt';
import { PdfColor } from '@syncfusion/ej2-pdf-export';
import { ClickEventArgs } from '@syncfusion/ej2-navigations/src/toolbar/toolbar';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        gantt.pdfExport();
    }
};
let pdfQueryTaskbarInfo: EmitType<Object> = (args: Object) => {
    if (args.data.Progress < 50 && !args.data.hasChildRecords) {
        args.taskbar.progressColor = new PdfColor(205, 92, 92);
        args.taskbar.taskColor = args.taskbar.taskBorderColor = new PdfColor(240, 128, 128);
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    columns: [
        { field: 'TaskID' },
        { field: 'TaskName', visible: false },

```



```

        { field: 'StartDate'},
        { field: 'Duration'},
        { field: 'Progress'}
    ],
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler,
    pdfQueryTaskbarInfo: pdfQueryTaskbarInfo
});
ganttt.appendTo('#GanttExport');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Theme

PDF export provides an option to include theme for the exported PDF document. To apply theme in exported PDF, define the **theme** in **pdfExportProperties**. The available themes are:

- Material
- Fabric
- Bootstrap
- Bootstrap 4

INDEX.TS

```
import { Gantt, Toolbar, PdfExport, Selection, PdfExportProperties } from
'@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
navigations/src/toolbar/toolbar';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        let exportProperties: PdfExportProperties = {
            theme: "Fabric"
        };
        gantt.pdfExport(exportProperties);
    }
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="GanttExport"></div>
```

```

        </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customized Theme

PDF export provides an option to customize the Gantt style for the exported PDF document. To customize Gantt style in exported PDF, define the `ganttStyle` in `pdfExportProperties`.

INDEX.TS

```

import { Gantt, Toolbar, PdfExport, Selection, PdfExportProperties } from
 '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-
 navigations/src/toolbar/toolbar';
import { PdfColor } from '@syncfusion/ej2-pdf-export';
import { PdfPaddings } from '@syncfusion/ej2-gantt/src/export/pdf-base/pdf-
 borders';
Gantt.Inject(Toolbar, PdfExport, Selection);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        let exportProperties: PdfExportProperties = {
            ganttStyle: {
                fontFamily: 1,
                columnHeader: {
                    backgroundColor: new PdfColor(179, 219, 255)
                },
                taskbar: {
                    taskColor: new PdfColor(240, 128, 128),
                    taskBorderColor: new PdfColor(240, 128, 128),
                    progressColor: new PdfColor(205, 92, 92)
                },
                connectorLineColor: new PdfColor(128, 0, 0),
                footer: {
                    backgroundColor: new PdfColor(205, 92, 92)
                },
                timeline: {
                    backgroundColor: new PdfColor(179, 219, 255),
                    fontStyle: 1
                },
                label: {
                    fontColor: new PdfColor(128, 0, 0)
                },
                cell: {
                    backgroundColor: new PdfColor(240, 248, 255),
                    fontColor: new PdfColor(0, 0, 0),
                    borderColor: new PdfColor(179, 219, 255)
                },
            },
        };
    }
}

```

```

    };
    gantt.pdfExport (exportProperties);
  }
};
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  columns: [
    { field: 'TaskID' },
    { field: 'TaskName', visible: false },
    { field: 'StartDate' },
    { field: 'Duration' },
    { field: 'Progress' }
  ],
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="GanttExport"></div>
  </div>

  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing header and footer of PDF export in EJ2 JavaScript Gantt control

PDF export provides an option to specify and customize text, page number, line and image in header and footer of exported PDF document by using [pdfExportProperties](#).

Write a text in header and footer

This functionality helps to customize the text that appears in the header or footer sections of a PDF document. Text can be added to [header](#) or [footer](#) of exported PDF document by using [pdfExportProperties](#).

- **type** property in the content array indicates the content type, such as 'Text'.
- **Value** property determines the text.
- **Position** property determines the horizontal and vertical positions of the text element.
- **style** property define the visual styling properties for the text element

`ts

```

let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'Text',
        value: 'INVOICE',
        position: { x: 380, y: 0 },
        style: { textBrushColor: '#C25050', fontSize: 25 },
      },
    ]
  },
};

```

Draw a line in header and footer

This functionality helps to customize the line that appears in the header or footer sections of the PDF document. A line can be added to [header](#) or [footer](#) of the exported PDF document by using [pdfExportProperties](#).

- `type` determines content type, such as 'Line'.
- `style` is used to set properties like the color (`penColor`), size (`penSize`), and style (`dashStyle`) of the line.
- `points` specifies the coordinates for the start and end points of the line.

Supported line styles:

- `dash`
- `dot`
- `dashdot`
- `dashdotdot`
- `solid`

```
`ts
```

```
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'Line',
        style: { penColor: '#000080', penSize: 2, dashStyle: 'Solid' },
        points: { x1: 0, y1: 4, x2: 685, y2: 4 }
      }
    ]
  }
}
```

Add page number in header and footer

This feature allows to customize the page number that appears in the header or footer sections of the PDF document. Page numbers can be added in [header](#) or [footer](#) of the exported PDF document by using [pdfExportProperties](#).

- `type` indicates that the content is a page number.
- `pageNumberType` specifies the type of numbering to be used.
- `format` is an optional attribute that allows you to customize the text format of the page number.
- `position` defines the coordinates (x, y) where the page number will be located.
- `style` sets the styling properties of the page number text, such as color (`textBrushColor`), font size (`fontSize`), and horizontal alignment (`hAlign`).

Supported page number types:

- LowerLatin - a, b, c,
- UpperLatin - A, B, C,
- LowerRoman - i, ii, iii,
- UpperRoman - I, II, III,
- Number - 1,2,3.

`ts

```
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'PageNumber',
        pageNumberType: 'Arabic',
        format: 'Page {$current} of {$total}', //optional
        position: { x: 0, y: 25 },
        style: { textBrushColor: '#ffff80', fontSize: 15, hAlign: 'Center' }
      }
    ]
  }
}
```

Insert an image in header and footer

This feature allows to customize the image that appears in the header or footer sections of the PDF document. Image (Base64 string) can be added in the exported document in [header](#) or [footer](#) of the exported PDF document by using [pdfExportProperties](#).

- **type** indicates that the content is an image.
- **src** specifies the source of the image, which should be Base64 string.
- **Position** determines the horizontal and vertical positions of the image will be located.
- **size** sets the dimensions of the image.

Note: PDF Export supports base64 string to export the images.

`ts

```
// Replace it with a valid Base64-encoded image.
```

```
let image: string = "/9j/4AAQSkZJRgABAQEAAeAB4AAD..."
```

```
let exportProperties: PdfExportProperties = {
```

```
header: {
```

```
fromTop: 0,
```

height: 130,

```
contents: [
```

 $\{$

```
type: 'Image',
```

src: image,

```
position: { x: 40, y: 10 },
```

```
size: { height: 100, width: 250 },
```

}

1

}

}

.

The below code illustrates the pdf export customization.

INDEX.TS

[illegible]

zQbLk+7vXx56/dn8kFtfPQXHDttSj+lb8XcuL+bxT7u9Oev3Z/JBbXz0FxbwUo/pW/F3JfzeKfd3
qcsGXCysvhPKtCizlkJ3Izs7rbVuSS2Farq2qZORqS1aNS0taaSERS1ERKLubt4ddW4rcmsBe8GO
IltgssU6CJXuo8vfvSh7ASnWUvWhyeRyD1s2ecmU6DdWW4W0W2fzEYpFdK2sq5LbFAIMw8bkcUcY
6BkGU6LSLXVarjqbj7JFFkFuM5zxR2nINNl1QiIh+LiHYuKeW8++s3HXFnVS1mdTUZ7ZmY+cIj3x
XuixCS5xtJOMk4yVvBjGwmhjBYALABiAzL5jquy+8FGxkti2Y+XxTsNEw6ycaeaVkrQotsjHrBjR
ZaK2NACWvabQRgIOj7w4jgXnFhQ48MworQ5pwEHEVsvdPfdB2tJmz9pltws6pktuUJLUX3var/wB
Og9rcLfdT6+QqZuZkKLGtGQ4mvlZner+23CBp+s1UIlGWzUmC6DlGMt15xp6c5lgbIVGue4Q2wPe
HvYmXq6xmUfwuFyh1rpf3h1KjFvpae9IbXONQgXIFygIrs8WF2Os23yxvBMigVo4aOSO1Scn4vnV
YE669zPw6I4VQvjN43UOpRhxrpijuiAisCxTvVF5veyn7yKfW3FB9rsWdJedzKwtWVQ8kyI6ZjMq5
xSzbZgWeNK08vVs5bGSWriou2C1RTse4pbUakj1J5O0SO1ySoWrtfPpP5grdRlKSFJpi0punRCSH
jeuGQDi2Dzcmc4zv6rc/R85ItZibkMFhblB057T52XRIWgirqfQEQUEU1XEXTHNX2LcWkhv0JlROS
6HWXtncPM6oulIy6Ets8+gs+1vB9U+/Httpfb+mMMNVGI886Ad7nOHFYteVzrNezXUbKHdnA85gf
NGK5cwwY8Wxg3ktSqPcIbYHvD3sTLldYzKP4XC5Q610i7w6lRi30tPekNrnGoQLKc5QEVPeLc7HW
bb5Y3gmRQK0cNHJHapOT8XzqsCdde5n4dEcKoXxm8bqHUow410x3RARWBYP3qi83vZT95FPrbig+
12LOkv05lYYKYs9edP7PyilEqeks8gkRMK+VFJVPSe0pJ6UqLaMs5DBpGjZWlpZ0pOMumOydoOQj
IRhCy5Gej0dHbMSzrlw+7DnByhasXn3Szi72KVGm5cbJHVe1RZFnaM9CHaaD3FaD+Y8w+dq11OmQ
tRDFba+XOJ2Vuh+bQcR0HAt1ldrPL04zY3bmMMbc+lufSMY1YVgIpy6ki5y6p23ky5KTZpaJFBL
o6ectdOF+ySe4Xvj/AJbea9VJqi6sUxfEyp8AjsOH1zxRoHndGeyolqrI2hYOWQD+s4YPVHGOnN0
ratppqHaQww2ltttJIQhJUSlJFQiIt0h9GMY2G0MYLAMAC0k5znuLnG0l1cx2XVR7hDmRXDXhmZ0I
rMTKp/wDGWMyjuFwuUotdIu8OpUVNx8Dqaf01j3Jfte/iNtGG+3EVbhwrzlr+B+OsecL8Rxs8xS
6GdNfwPx1jzhfiGxvzFLoZ1ahivnWnsHOBzLdQ4nlljSqrLgXSWdwa/rSCJ4A8UdqlJM2w8GdVdz
uOgSnkzI4xgjKoiCMjcLN7aoX6Gx1w3BkHUowuFpwrp6/gfjrHnC/EdtjfmK4uhnTX8D8dY84X4h
sb8xS6GdWDYpiIYfiLztQfbcySlNchRHTqncFOrC0tEG0cbsWfIkG6s0Kw8UtSCAi+MXCQsfDOWc
bDtvS PJNDjbiSulaT0kZHPhnGgw5iGYUvOLTgIOEELvCivgvESGbhDCCMYUFWHwaSetRDqs9Mnby
KJcNUShaquwidJpbr7oj0JrnTtloN0014LLufbeMS5l3Hda75gzNzg4m273LaMC2ZI+EC5knCbZb
GaMGZ2k5rMZsx5LFN0nlEukEsh5PKYVMPCQqCbabToIv/AEz0me6NuSulAo6XZKyzbljRYAFriam
o07GdMR3WucbSV3RlLHQEXkWunUhs5Zabz+1GTyHl0E9FR+Uybx3Qg1OVQRGayySPoSI66KD0gs
fFiNZDxk4Na4cQBaVq6nDjwD1ESiiGaGVS/UaN/LCwbX6XzfG35li31A+we5fvNxyCHw7P1Hjfyw
4/AKWzfG35kvqB9g9yc3FgIfDs/UeN/LB+Autm+NvzJfUD7B7l0tyF6dlF7Vj37U3PuIVI2Y1yEc
NMrdgKPoSk1+1uIQo8yk56UMRM9KTELf2OZ31luMHBrBKyIb2vFrMSg6Kw3MBaHin4eJfZlZl1bb
n6kxiujSoyVnKGz5yPOJUUDSxAlHxt+ZeBmYANh6j3L583FgIfDs/UeN/LDn8ApbN8bfmF9QPsH
uTm4sBD4dn6jxv5YPwC1s3xt+ZL6gfYPcpauDvyuGvlVOyuTcbUcp1DklkyN6XU1TL1LprSMv3C9
FafNUR1IUfNyNzfFWW2zCD1Er2hRWRN4sknf9d3chmsVJpnN324uDcNp5CYN5ZJUW1UkmR/wAhr2e
r3QVGzL5SYikPYbCAx5sOsNIVqlao0tOwGzEGGC1wtG6aMH0V0+aBut7uRH9A/wD4DF/Mirnpnf2
4nyrI2kU16Ifub3pzQN1vdyI/oH/8A/Mirnpnf24nyptIpr0Q/c3vXr2WvWsRbKaHJrPzN1+LJpT
2QuFdbLISZEZ1Uki98Qk6IrhRFOTF6yMQufYTYWubgFluFwAyhYFJVapKiYN8TTAG22b5pwnUTmW
XizqBQEQUEu4Q2wPeHvYmXq6xmUfwuFyh1rpf3h1KjFvpae9IbXONQgXIFygIrs8WF2Os23yxvBM
igVo4aOSO1Scn4vnVYE669zPw6I4VQvjN43UOpRhxrpijuiAisCxTvVF5veyn7yKfW3FB9rsWdJed
zKQL1Nkm0nh6/sIfEtbvL85y/9QvpKrfkiW5I7ViorymkBFKODhskH4tiP7mxsHwY+X/6b/8AJip
dffJHtt6nLaUfQ60sgIgIo9whtge8PexMv1jMo/hcLlDrXSLvDqVGLfS096Q2ucahAuQLlARWl4
sLsdZtv1jeCZFArRw0ckdqk5PxfOqwJ117mfh0RwqhfGbxuodSjDjXTHdEBFYfineqLze9lP3kU+
tuKD7XYs6S87mUgXqbJNpPD1/YQ+Ja3eX5zl/6hfSVW/JETyR2rFRXlNICKUcHDZIPxBeF3NjYPg
x8v8A9N/+TFS6++SPbb1OW0o+hlpZARARR7hDbA94e9izerrGZR/C4XKHwukXeHUqMW+lp70htc4
1CBcgXKAitLxYXY6zbfLG8EyKBWjho5I7VJyfi+dVgTrr3M/DojhVC+M3jdQ6lGHGumO6ICKwLFO
9UXm97KfvIp9bcUH2uxZ0153MpAvU2SbSeHr+wh8Slu8vznL/ANQvpKrfkiW5I7ViorymkBFKODh
skH4tiP7mxsHwY+X/AOm//JipdffJHtt6nLaUfQ60sgIgIvJtZCWbj7LzeCtjrXkE/BPNzPXTmps
61NBk7lqqWsnJrU6lQh6Qi9sRph763BrXBsIw4lrCV2GLXpmK62nj9HHCEvqnPX6D3LGUJbQnsYY
tfcut9Po44cXlTnr9H0S5l9Cexhi19y630+jjgvqnPX6PolzL6FONycjuRfKfYiAuI5Acrpxri3+
Q0YUQU4aU5dVepVFZORUq7gip58lEiaZdt12v2L2hhgG4xKE4q7HFvnFPkiyuv1c3FG7lT5BKy
6nlV9u0lRUSgmqbsF13Zq+i8SyX0L5exhi19y630+jjhzfVOev0fRLmX0J7GGLX3LrfT6OOC+qc9
fo+iXMv0Us3DWXwabOKnXM9cquVEFD8luQcwTENmy9R1Si1Z0lymiucR8/Fnolzft11stFmtesIQ
xbsdi69rJ7grQtpJjD2ytbYiGnjbxlHMxk5aafQ7QjotBuEaToZZjIVWYqFRlJxTNxpIPc/CXWE2
qegVppKThtgQpgta3ABgwe5eTyyYGP773eenmeMHj+WlEfy8ftK9duVLfxR93cnLJgY/vvd56eZ4
wPy0oj+Xj9pTb1S38Ufd3LKLuz1g6xto9bXX2mslHTto4datRlc0biHzZI05Z5CVmeSR5NTptkMuS
qdIUDEvqWlBCdZzBYRgOTDqWLOVhn6Uh7BMxi9tttMDGPREootARARYBhA7BN4e9aaequDLo/hcL
lDrXSJvDqVFraEamnoE+5LaG2ScKhAuWQjtS8g4XKZCO1LyAitKxYBEWDxOiIiIuWam0fwGBQa0c
NbyR1lSun4vnVYU8Qjk9NTyS64RO19KoXxh3DdQ6lGnGV0shHal5B2RMhHal5ARb/AOKdIijryyI

```

iL2uV6P8AeIFPrbig+12LOkv05lq1hcISeFDeeZpKpz9e19C0LBRB/wCvg8ntKxY/jXa+xRLkI7U
vIJBeaZCO1LyAi2rxZ6UlhOkZJIv1Zm019LDiv1n4B7Q6ismT8bzdytkGvFKICICLAMIHYJvD3rT
T1VwZdH8Lhcoda6RN4dSoub6WnvSG2DjUIFYBcoCK0nFgdjxOt88ZwDAoNaOGt5I6ypKT8XzqsOe
dfZr4wieFUL3D3jdQ6lGnGukO6ICLf/FPdXXlfw5X9sQKfWzFC9rsWdJedzLVrC47KC8/x+vgWhP
0R5Pg8ntKxY/jXa+xRKJFeaAi2rxZ/ZOFvZmPCw4r9ZuAe0OorJk/G83crYxrxSiAiAiWDCB2Cbw
96009VcGXR/C4XKHwukTeHUqLm+lp70htg41CBcgXKAitJxYHY8TrfPGcAwKDWjhreS0sqSk/F86
rDnnX2a+MInhVC9w943UOpRpxrpDuiAi3/wAU9ldeV/Dlf2xAp9bMUL2uxZ0153MtWsLjsoLz/H6
+BaE/RHk+Dye0rFj+Ndr7FEokV5oCLavFn9k4W9mY8LDiv1m4B7Q6ismT8bzdytkGvFKICICLybW
2Z11tLLTiyE3U8mBncC/L4k2VZLhNOoNCsk6HQ6KOhj0hRXQYjYjcYNvQuHC6FhWrScWDg5pSSSm
VsqEVOuiOKE/tonszeH15w9K/edhYondK2PpRHFbtonszej6pecPSnOwsHTulbH0ojig20T2ZvR
9UvOHpU63H3E2LuAsfE2Js09MnJfFRrketUe+TrmqRqHJ0USSzUQWam6Imen4tIRRFjWW2WYF7Q4
bYQuWqDorFLY08ZFPxj0xthqkQ6t5eTM0EWUpRqOhanmKpmJUVnnWgABuDQvG84elfPnYWDp3Stj
6URxQ520T2ZvR9UvOHpTnYWDp3Stj6URxQbaJ7M3o+qXnD0qWrgsFy7fBxcnbtgYicvKnxMFFckY
snqEz15OTRKadMOunaEdSFKR6Sudms3NtlgsxrlhQWwrbLYLeDi+bjLyrctTu8C0MfahMzn8WcZF1
DTBDbRLNJJ6FOpnQqJLbGXL1hm5aE2CyyxosGBebpWG9xccqx/nYWDp3Stj6URxQ9ttE9mb0fVcX
nD0pzsLB07pWx9KI4oNtE9mb0fVLzh6VntyebVdDcLbfl/sTGWhdmesnoCkdHJdalJxSDV0JILPV
tNDrujDnqbmZ+FsmWyy23AM3/AKu8OXZCddNU+iIXugIgIgIgIgIgIgIgIgIgIgIgIgIgIgIgIgI
gIv/Z';
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'GanttExport_pdfexport') {
        let exportProperties: PdfExportProperties = {
            header: {
                fromTop: 0,
                height: 150,
                contents: [
                    {
                        type: 'Text',
                        value: 'Welcome',
                        position: { x: 380, y: 0 },
                        style: { textBrushColor: '#C25050', fontSize: 25 },
                    },
                    {
                        type: 'Image',
                        src: image,
                        position: { x: 400, y: 70 },
                        size: { height: 50, width: 50 },
                    },
                ],
            },
            footer: {
                fromBottom: 160,
                height: 100,
                contents: [
                    {
                        type: 'Text',
                        value: 'Thank you!',
                        position: { x: 350, y: 40 },
                        style: { textBrushColor: '#C67878', fontSize: 14 },
                    },
                    {
                        type: 'PageNumber',
                        pageNumberType: 'Arabic',
                        format: 'Page { $current } of { $total }',
                        position: { x: 0, y: 25 },
                        size: { height: 50, width: 100 },
                        style: { textBrushColor: '#000000', hAlign:
'Center', vAlign: 'Bottom' }

```

```

    }
    ],
    },
    };
    gantt.pdfExport(exportProperties);
}
};
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    toolbarClick: clickHandler
});
gantt.appendTo('#GanttExport');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css" rel="stylesheet"
type="text/css">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="GanttExport"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Data markers in EJ2 JavaScript Gantt control

Data markers are a set of events used to represent the schedule events for a task. Data markers are defined in data source as array of objects, and this value is mapped to the Gantt control using the [taskFields.indicators](#) property. You can represent more than one data marker in a task.

Data markers can be defined using the following properties:

- [date](#): Defines the date of indicator.
- [iconClass](#): Defines the icon class of indicator.
- [name](#): Defines the name of indicator.
- [tooltip](#): Defines the tooltip of indicator.

Note: Data Marker **tooltip** will be rendered only if tooltip property has value.

The following code example demonstrates how to implement data markers in the Gantt chart.

INDEX.TS

```
import { Gantt } from '@syncfusion/ej2-gantt';
let gantt: Gantt = new Gantt({
  dataSource: [
    {
      TaskID: 1,
      TaskName: 'Project Initiation',
      StartDate: new Date('04/02/2019'),
      EndDate: new Date('04/21/2019'),
      subtasks: [
        {
          TaskID: 2, TaskName: 'Identify Site location',
          StartDate: new Date('04/02/2019'), Duration: 4, Progress: 50,
          Indicators: [
            {
              'date': '04/08/2019',
              'iconClass': 'e-btn-icon e-notes-info e-icons e-
icon-left e-gantt e-notes-info::before',
              'name': 'Custom String',
              'tooltip': 'Follow up'
            },
            {
              'date': '04/11/2019',
              'iconClass': 'e-btn-icon e-notes-info e-icons e-
icon-left e-gantt e-notes-info::before',
              'name': '<span style="color:red">String
Template</span>',
            }
          ]
        },
        { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
        {
          TaskID: 4, TaskName: 'Soil test approval', StartDate:
new Date('04/02/2019'), Duration: 4, Progress: 50, },
      ]
    },
  ],
});
```

```

    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            {
                TaskID: 6, TaskName: 'Develop floor plan for
estimation', StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50,
                Indicators: [
                    {
                        'date': '04/10/2019',
                        'iconClass': 'e-btn-icon e-notes-info e-icons e-
icon-left e-gantt e-notes-info::before',
                        'name': 'Indicator title',
                        'tooltip': 'tooltip'
                    }
                ]
            },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 }
        ]
    },
    ],
    height: '450px',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        resourceInfo: 'resources',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks',
        indicators: 'Indicators'
    },
    });
ganttt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <style>
        .e-gantt .e-gantt-chart .e-custom-event-marker {
```

```

        width: 1px;
        border-left: 2px green dotted;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Global local in EJ2 JavaScript Gantt control

Localization

The [Localization](#) library allows you to localize default text content of the Gantt. The Gantt component has static text on some features (like toolbar area text, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the

[locale](#) value and translation object.

The following list of properties and its values are used in the Gantt.

Locale key words | Text

emptyRecord | No records to display

id | ID

name | Name

startDate | Start Date

endDate | End Date

duration | Duration

progress | Progress

dependency | Dependency

notes | Notes

baselineStartDate | Baseline Start Date

baselineEndDate | Baseline End Date

type | Type

offset | Offset

resourceName | Resources

resourceID | Resource ID

day | day

hour | hour

minute | minute

days | days

hours | hours

minutes | minutes

generalTab | General

customTab | Custom Columns

writeNotes | Write Notes

addDialogTitle | New Task

editDialogTitle | Task Information

add | Add

edit | Edit

update | Update

delete | Delete

cancel | Cancel

search | Search

task | task

tasks | tasks

zoomIn | Zoom in

zoomOut | Zoom out

zoomToFit | Zoom to fit

expandAll | Expand all

collapseAll | Collapse all

nextTimeSpan | Next timespan

prevTimeSpan | Previous timespan

saveButton | Save

taskBeforePredecessor_FS | You moved "{0}" to start before "{1}" finishes and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskAfterPredecessor_FS | You moved "{0}" away from "{1}" and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskBeforePredecessor_SS | You moved "{0}" to start before "{1}" starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskAfterPredecessor_SS | You moved "{0}" to start after "{1}" starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskBeforePredecessor_FF | You moved "{0}" to finish before "{1}" finishes and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskAfterPredecessor_FF | You moved "{0}" to finish after "{1}" finishes and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskBeforePredecessor_SF | You moved "{0}" away from "{1}" to starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

taskAfterPredecessor_SF | You moved "{0}" to finish after "{1}" starts and the two tasks are linked. As the result, the links cannot be honored. Select one action below to perform

okText | Ok

confirmDelete | Are you sure you want to Delete Record?

from | From

to | To

taskLink | Task Link

lag | Lag

start | Start

finish | Finish

enterValue | Enter the value

taskInformation | Task Information

deleteTask | Delete Task

deleteDependency | Delete Dependency

convert | Convert

save | Save

above | Above

below | Below

child | Child

milestone | Milestone

toTask | To Task

toMilestone | To Milestone

eventMarkers | Event markers

leftTaskLabel | Left task label

rightTaskLabel | Right task label

timelineCell | Timeline cell

confirmPredecessorDelete | Are you sure you want to remove dependency link?

taskMode | Task Mode

changeScheduleMode | Change Schedule Mode

subTasksStartDate | SubTasks Start Date

subTasksEndDate | SubTasks End Date

scheduleStartDate | Schedule Start Date

scheduleEndDate | Schedule End Date

auto | Auto

manual | Manual

zoomToFit | Zoom to fit

excelExport | Excel export

csvExport | CSV export

pdfExport | Pdf export

unit | Unit

work | Work

taskType | Task Type

unassignedTask | Unassigned Task

group | Group

Loading translations

To load translation object in an application use [load](#) function of [L10n](#) class.

The below example demonstrates the Gantt in **Deutsch** culture.

INDEX.TS

```
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
setCulture('de-DE');
L10n.load({
  'de-DE': {
    'gantt': {
      "id": "Ich würde",
      "name": "Name",
      "startDate": "Anfangsdatum",
      "duration": "Dauer",
      "progress": "Fortschritt",
    }
  }
});
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
```

```

    locale: 'de-DE',
    taskFields: {
      id: 'TaskID',
      name: 'TaskName',
      startDate: 'StartDate',
      duration: 'Duration',
      progress: 'Progress',
      child: 'subtasks'
    },
  });
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <style>
.e-gantt .e-gantt-chart .e-custom-event-marker {
  width: 1px;
  border-left: 2px green dotted;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Internationalization

The [Internationalization](#) library is used to globalize number, date, and time values in gantt component.

INDEX.TS

```

import { L10n, loadCldr, setCulture } from '@syncfusion/ej2-base';
import * as cagregorian from './ca-gregorian.js';
import * as numbers from './numbers.js';
import { Gantt } from '@syncfusion/ej2-gantt';
let GanttData: Object[] = [
    {
        TaskID: 1,
        TaskName: 'Project Initiation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 2, TaskName: 'Identify Site location', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 3, TaskName: 'Perform Soil test', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
            { TaskID: 4, TaskName: 'Soil test approval', StartDate: new
Date('04/02/2019'), Duration: 4, Progress: 50 },
        ]
    },
    {
        TaskID: 5,
        TaskName: 'Project Estimation',
        StartDate: new Date('04/02/2019'),
        EndDate: new Date('04/21/2019'),
        subtasks: [
            { TaskID: 6, TaskName: 'Develop floor plan for estimation',
StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 7, TaskName: 'List materials', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 },
            { TaskID: 8, TaskName: 'Estimation approval', StartDate: new
Date('04/04/2019'), Duration: 3, Progress: 50 }
        ]
    }
]
loadCldr(cagregorian,numbers);
setCulture('de-DE');

L10n.load({
    'de-DE': {
        'gantt': {
            "id": "Ich würde",
            "name": "Name",
            "startDate": "Anfangsdatum",
            "duration": "Dauer",
            "progress": "Fortschritt",
        }
    }
});
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    height: '450px',
    locale: 'de-DE',
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
    }
});

```

```

        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <style>
        .e-gantt .e-gantt-chart .e-custom-event-marker {
            width: 1px;
            border-left: 2px green dotted;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* In the above sample, **Timeline** is formatted by **NumberFormatOptions** and **DateFormatOptions**.

* By default, [locale](#) value is **en-US**. If you want to change **en-US** culture, then set the [locale](#).

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Gantt component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Urdu, etc.). To enable RTL Gantt, set the [enableRtl](#) to true.

INDEX.TS

```
import { L10n, loadCldr, setCulture } from '@syncfusion/ej2-base';
import { Gantt, Toolbar } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar);
setCulture('ar-AE');
L10n.load({
  'ar-AE': {
    "gantt": {
      "emptyRecord": "لا سجلات لعرضها",
      "id": "هوية شخصية",
      "name": "اسم",
      "startDate": "تاريخ البدء",
      "endDate": "تاريخ الانتهاء",
      "duration": "المدة الزمنية",
      "progress": "تقدم",
      "dependency": "الاعتماد",
      "notes": "ملاحظات",
      "baselineStartDate": "تاريخ البدء الأساسي",
      "baselineEndDate": "تاريخ نهاية خط الأساس",
      "taskMode": "وضع المهام",
      "changeScheduleMode": "تغيير وضع الجدول",
      "subTasksStartDate": "تاريخ بدء المهام الفرعية",
      "subTasksEndDate": "تاريخ انتهاء المهام الفرعية",
      "scheduleStartDate": "جدولة تاريخ البدء",
      "scheduleEndDate": "تاريخ انتهاء الجدول الزمني",
      "auto": "تلقائي",
      "manual": "كاتب",
      "type": "اكتب",
      "offset": "عوض",
      "resourceName": "مصادر",
      "resourceID": "معرف المورد",
      "day": "يوم",
      "hour": "ساعة",
      "minute": "دقيقة",
      "days": "أيام",
      "hours": "ساعات",
      "minutes": "الدقائق",
      "generalTab": "جنرال لواء",
      "customTab": "أعمدة مخصصة",
      "writeNotes": "اكتب ملاحظات",
      "addDialogTitle": "مهمة جديدة",
      "editDialogTitle": "معلومات المهمة",
      "saveButton": "حفظ",
      "add": "إضافة",
      "edit": "تعديل",
      "update": "تحديث",
      "delete": "حذف",
      "cancel": "إلغاء",
      "search": "بحث",
      "task": "مهمة",
    }
  }
});
```

```

"tasks": "مهام",
"zoomIn": "تكبير",
"zoomOut": "تصغير",
"zoomToFit": "تكبير لتناسب",
"excelExport": "اكسل التصدير",
"csvExport": "تصدير CSV",
"expandAll": "توسيع الكل",
"collapseAll": "انهيار جميع",
"nextTimeSpan": "الجدول الزمني التالي",
"prevTimeSpan": "الجدول الزمني السابق",
"okText": "حسنًا",
"confirmDelete": "هل أنت متأكد أنك تريد حذف السجل؟",
"from": "من عند",
"to": "إلى",
"taskLink": "رابط المهمة",
"lag": "بطئ",
"start": "بداية",
"finish": "إنهاء",
"enterValue": "أدخل القيمة",
"taskBeforePredecessor_FS": "0 '{قبل بنقل}' للبدء قبل انتهاء",
    ويتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء '{1}'
    واحدًا أدناه للقيام به",
"taskAfterPredecessor_FS": "0 '{بعيدًا عن}' 1 '{ويتم}'",
    وربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا أدناه
    للقيام به",
"taskBeforePredecessor_SS": "0 '{قبل بنقل}' للبدء قبل أن يبدأ",
    وربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا '{1}'
    أدناه للقيام به",
"taskAfterPredecessor_SS": "0 '{قبل بنقل}' للبدء بعد بدء 1 '{و}'",
    وربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا أدناه
    للقيام به",
"taskBeforePredecessor_FF": "0 '{قبل بنقل}' للإنهاء قبل انتهاء",
    ويتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء '{1}'
    واحدًا أدناه للقيام به",
"taskAfterPredecessor_FF": "0 '{قبل بنقل}' للإنهاء بعد انتهاء",
    ويتم ربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء '{1}'
    واحدًا أدناه للقيام به",
"taskBeforePredecessor_SF": "0 '{قبل بنقل}' 1 '{بعيدًا عن}' للبدء",
    التشغيل وترتبط المهمتان. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء
    واحدًا أدناه للقيام به",
"taskAfterPredecessor_SF": "0 '{قبل بنقل}' 1 '{بعيدًا عن}' للإنهاء بعد بدء",
    وربط المهمتين. ونتيجة لذلك ، لا يمكن احترام الروابط. حدد إجراء واحدًا أدناه
    للقيام به",
"taskInformation": "معلومات المهمة",
"deleteTask": "حذف المهمة",
"deleteDependency": "حذف التبعية",
"convert": "تحويل",
"save": "حفظ",
"above": "في الأعلى",
"below": "أدناه",
"child": "طفل",
"milestone": "معلما",
"toTask": "لمهمة",
"toMilestone": "إلى معلم",
"eventMarkers": "علامات الحدث",
"leftTaskLabel": "تسمية المهمة اليسرى",
"rightTaskLabel": "تسمية المهمة الصحيحة",

```

```

        "timelineCell": "خلية الجدول الزمني",
        "confirmPredecessorDelete": "هل أنت متأكد أنك تريد إزالة رابط",
        "التبعية؟",
        "unit": "وحدة",
        "work": "عمل",
        "taskType": "نوع المهمة",
        "unassignedTask": "مهمة غير محددة",
        "group": "مجموعة",
        "indent": "مسافة بادئة",
        "outdent": "عفا عليها الزمن",
        "segments": "شرائح",
        "splitTask": "تقسيم المهمة",
        "mergeTask": "مهمة الدمج",
        "left": "اليسار",
        "right": "حق"
    }
}
});
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    locale: 'ar-AE',
    enableRtl: true,
    height: '450px',
    toolbar: ['ExpandAll', 'CollapseAll'],
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <style>
    .e-gantt .e-gantt-chart .e-custom-event-marker {
        width: 1px;
        border-left: 2px green dotted;
    }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Internationalization](#)
- [Localization](#)

Accessibility in EJ2 JavaScript Gantt control

The Gantt component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Gantt component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation |

|

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Gantt component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Gantt component:

The following ARIA attributes are used in Gantt:

| Attributes | Purpose |

| --- | --- |

| **grid (role)** | This attribute is added to the **e-table** element present in the Gantt, which represents Grid part |

| **gridcell (role)** | This attribute is added to the **td** elements present within the **e-table**, which represents the work cells of Gantt |

| **columnheader (role)** | This attribute is added to the **th** elements within the **e-table**, which represents the header cells of Grid table |

| **separator (role)** | This attribute is added to the **e-split-bar** element, which represents the splitter between the Grid table and Chart |

| **dialog (role)** | This attribute is added to the **e-dialog** element, which represents the pop-up dialog |

| **toolbar (role)** | This attribute is added to the **e-gantt-toolbar** element, which represents the toolbars of Gantt |

| **aria-label** | It indicates the element's information
It is assigned to the Gantt UI elements such as timeline cell, taskbar, left label, right label, dependency line, and event markers. |

| **aria-selected** | This attribute is assigned to the Gantt chart row, and it defaults to **false**. The value is changed to **true** when the user selects a grid cell or task |

| **aria-expanded** | This attribute is assigned to the Gantt chart parent task row. The value is changed to **true** when the user clicks a parent taskbar to expand. After the user clicked a parent taskbar to collapse, the attribute value is changed to **false** |

| **aria-grabbed** | This attribute is assigned to the taskbars of Gantt when the user tries to achieve taskbar editing |

Keyboard navigation

The Gantt component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Gantt component.

| **Press** | **To do this** |

| --- | --- |

| **Home** | Selects the first row. |

| **End** | Selects the last row. |

| **DownArrow** | Moves the cell focus/row or cell selection downward. |

| **UpArrow** | Moves the cell focus/row or cell selection upward. |

| **LeftArrow** | Moves the cell focus/row or cell selection left side. |

| **RightArrow** | Moves the cell focus/row or cell selection right side. |

| **Ctrl + Up Arrow** | Collapses all tasks. |

| **Ctrl + Down Arrow** | Expands all tasks. |

| **Ctrl + Shift + Up Arrow** | Collapses the selected row. |

| **Ctrl + Shift + Down Arrow** | Expands the selected row. |

| **Enter** | Saves request. |

| **Esc** | Cancels request. |

| **Insert** | Adds a new row. |

| **Ctrl + Insert** | Opens addRowDialog. |

| **Ctrl + F2** | Opens editRowDialog. |

| **Delete** | Deletes the selected row. |

| **Shift + F5** | FocusTask |

| **Ctrl + Shift + F** | Focus search |

| **Shift + DownArrow** | Extends the row/cell selection downwards. |

| **Shift + UpArrow** | Extends the row/cell selection upwards. |

| **Shift + LeftArrow** | Extends the cell selection to the left side. |

| Shift + RightArrow | Extends the cell selection to the right side. |

| Tab / Shift + Tab | To focus the close icon in the message. |

| Alt + j | Focus Gantt component. |

Ensuring accessibility

The Gantt component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Gantt component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Gantt component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

Touch interaction in EJ2 JavaScript Gantt control

The Gantt control supports to perform user interactions in mobile and tablet devices. This section explains how to interact with the Gantt features in touch-enabled devices.

Tooltip

To perform **touch and hold** action on a element, refer to [tooltip popup](#).

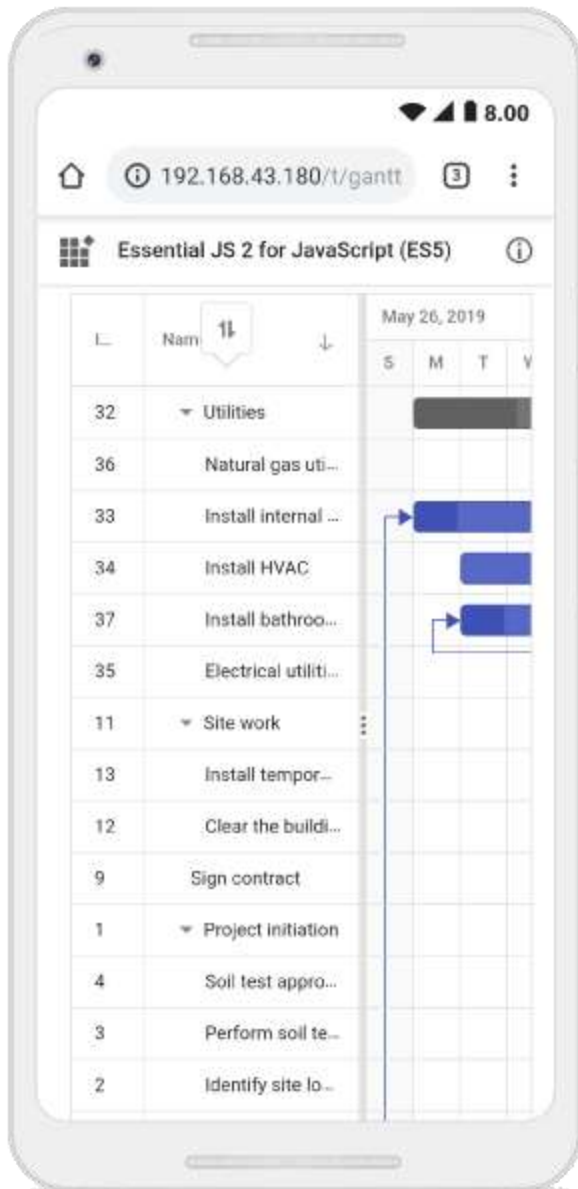
Context menu

To perform **long press** action on a row, [context menu](#) is opened, and then tap a menu item to trigger its action.

Sorting

To perform **tap** action on a column header, trigger [sorting](#) operation to the selected column. A popup is displayed for multi-column sorting. To sort multiple columns, tap the popup, and then tap the desired column headers.

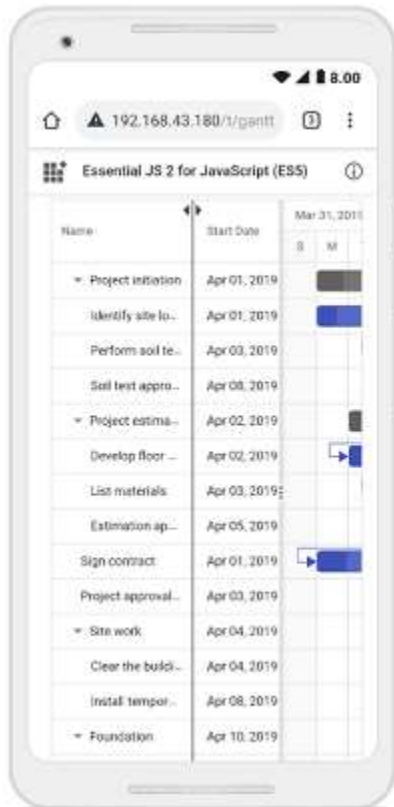
The following screenshot shows Gantt touch sorting,



Column resize

When the right edge of the column header cell is **tapped**, a floating handler will be visible over the right border of the column. To [resize](#) the column, drag the floating handler as needed.

The following screenshot represents the Gantt column resizing in touch device.



Editing

The Gantt control editing actions can be achieved using the double tap and tap and drag actions on a element.

The following table describes different types of editing modes available in Gantt.

Action | Description

Parent taskbar | You cannot create dependency relationship to parent tasks.



Taskbar without dependency | If you tap a valid child taskbar, it will create FS type dependency line between tasks, otherwise exits from task dependency edit mode.

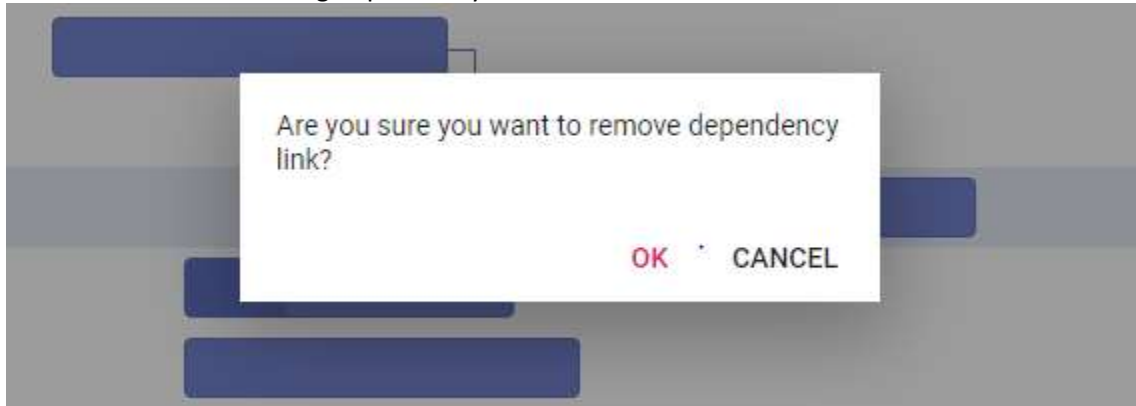


Taskbar with dependency | If you tap the second taskbar, which has already been directly connected,



it will ask to remove it.

Removing dependency | Once you tap the taskbar with direct dependency, then confirmation dialog will be shown for removing dependency.



INDEX.TS

```
import { Gantt, Edit, Selection } from '@syncfusion/ej2-gantt';
Gantt.Inject(Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: [
    {
      TaskID: 1,
      TaskName: 'Project Initiation',
      StartDate: new Date('04/02/2019'),
      EndDate: new Date('04/21/2019'),
      subtasks: [
        { TaskID: 2, TaskName: 'Identify Site location',
          StartDate: new Date('04/02/2019'), Duration: 3, Progress: 50 },
        { TaskID: 3, TaskName: 'Perform Soil test', StartDate:
          new Date('04/02/2019'), Duration: 4, Progress: 50 },
        { TaskID: 4, TaskName: 'Soil test approval', StartDate:
          new Date('04/02/2019'), Duration: 4, Predecessor: "2FS", Progress: 50 },
      ]
    },
    {
      TaskID: 5,
      TaskName: 'Project Estimation',
      StartDate: new Date('04/02/2019'),
      EndDate: new Date('04/21/2019'),
      subtasks: [
        { TaskID: 6, TaskName: 'Develop floor plan for
          estimation', StartDate: new Date('04/04/2019'), Duration: 3, Progress: 50 },
        { TaskID: 7, TaskName: 'List materials', StartDate: new
          Date('04/04/2019'), Duration: 3, Progress: 50 },
        { TaskID: 8, TaskName: 'Estimation approval', StartDate:
          new Date('04/04/2019'), Duration: 4, Predecessor: "6SS", Progress: 50 }
      ]
    },
  ],
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
```

```

        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    editSettings: {
        allowTaskbarEditing: true
    },
    // Forcing desktop layout to change as mobile layout
    load: function() {
        this.isAdaptive = true;
    }
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

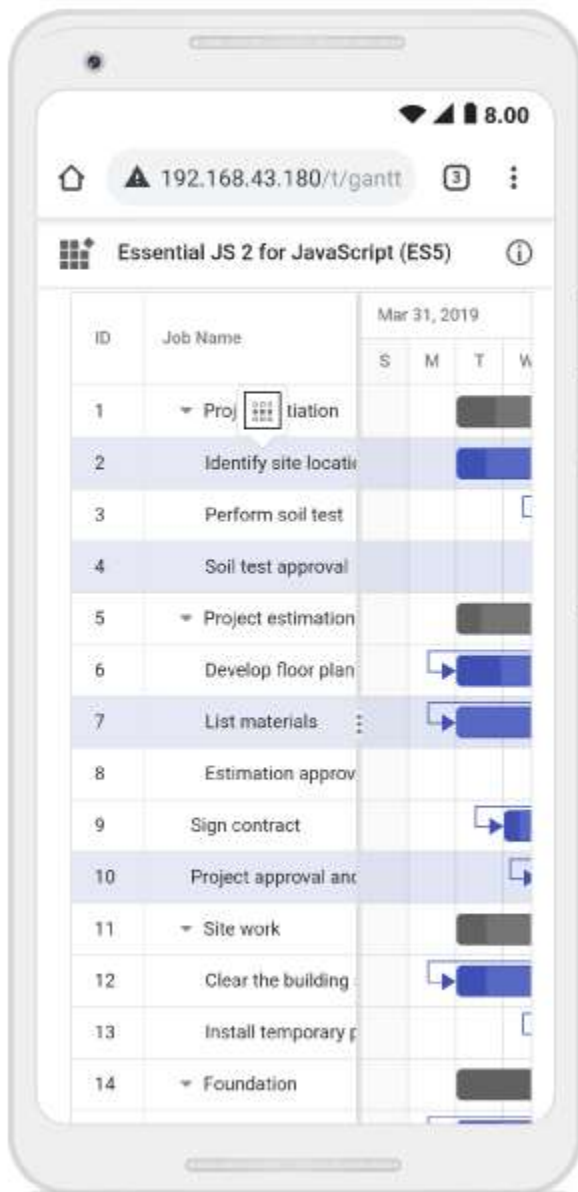
Note: In mobile device, you cannot create dependency other than FS by taskbar editing. By using cell/dialog editing, you can add all type of dependencies.

Selection

When you tap gantt row, tapped row will be selected.

[Single selection](#) : To select a single row or cell, perform single tap on it.

[Multiple selection](#) : To perform multiple selection, tap on the multiple selection popup, and then tap the desired rows or cells.



Ej1 api migration in EJ2 JavaScript Gantt control

This topic shows the API equivalent of JS2 Gantt component to be used, while migrating your project that uses JS1 Gantt.

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| ----- | ----- | ----- |

| Data Binding | **Property:** `dataSource`

`$("#gantt").ejGantt({
 dataSource: taskDetails,
});` | **Property:** `dataSource`


```
</></>var ganttObj = new ej.gantt.Gantt({<br>&#160;&#160;dataSource:
GanttData,<br>});<br>ganttObj.appendTo('#gantt');|
```

| To map id of task from data source | **Property:** *taskIdMapping*

 \$("#gantt").ejGantt({
taskIdMapping: 'taskId'
}); | **Property:** *taskFields.id*

 var ganttObj = new ej.gantt.Gantt({
taskFields: {
 id:
 'taskId'
}
});
ganttObj.appendTo('#gantt');|

| To map name of task from data source | **Property:** *taskNameMapping*

 \$("#gantt").ejGantt({
taskNameMapping: 'taskName'
}); | **Property:** *taskFields.name*

 var ganttObj = new ej.gantt.Gantt({
taskFields: {
 name:
 'taskName'
}
});
ganttObj.appendTo('#gantt');|

| To map start date from data source | **Property:** *startDateMapping*

 \$("#gantt").ejGantt({
startDateMapping: 'startDate'
}); | **Property:** *taskFields.startDate*

 var ganttObj = new ej.gantt.Gantt({
taskFields: {
 startDate:
 'startDate'
}
});
ganttObj.appendTo('#gantt');|

| To map end date from data source | **Property:** *endDateMapping*

 \$("#gantt").ejGantt({
endDateMapping: 'endDate'
}); | **Property:** *taskFields.endDate*

 var ganttObj = new ej.gantt.Gantt({
taskFields: {
 endDate:
 'endDate'
}
});
ganttObj.appendTo('#gantt');|

| To map duration from data source | **Property:** *durationMapping*

 \$("#gantt").ejGantt({
durationMapping: 'duration'
}); | **Property:** *taskFields.duration*

 var ganttObj = new ej.gantt.Gantt({
taskFields: {
 duration:
 'duration'
}
});
ganttObj.appendTo('#gantt');|

| To map duration unit from data source | **Property:** *durationUnitMapping*

 \$("#gantt").ejGantt({
durationUnitMapping: 'durationUnit'
}); | **Property:**
taskFields.durationUnit

 var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
 durationUnit: 'durationUnit'
}
});
ganttObj.appendTo('#gantt');|

| To map predecessors from data source | **Property:** *predecessorMapping*

 \$("#gantt").ejGantt({
predecessorMapping: 'predecessor'
}); | **Property:**
taskFields.dependency

 var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
 dependency: 'predecessor'
}
});
ganttObj.appendTo('#gantt');|

| To map progress from data source | **Property:** *progressMapping*

 \$("#gantt").ejGantt({
progressMapping: 'progress'
}); | **Property:** *taskFields.progress*

 var ganttObj = new ej.gantt.Gantt({
taskFields: {
 progress:
 'progress'
}
});
ganttObj.appendTo('#gantt');|

| To map child task from data source | **Property:** *childMapping*

 \$("#gantt").ejGantt({
childMapping: 'subTasks'
}); | **Property:** *taskFields.child*

 var ganttObj = new ej.gantt.Gantt({
taskFields: {
 child:
 'subTasks'
}
});
ganttObj.appendTo('#gantt');|

| To map baseline start date from data source | **Property:** *baselineStartDateMapping*

 \$("#gantt").ejGantt({
baselineStartDateMapping: 'baselineStartDate'
}); | **Property:**

```
taskFields.baselineStartDate <br/><br/>var ganttObj = new ej.gantt.Gantt({<br/>taskFields:
{<br/>&#160;&#160;baselineStartDate:
'baselineStartDate'<br/>}<br/>});<br/>ganttObj.appendTo('#gantt');|
```

| To map baseline end date from data source | **Property:** *baselineEndDateMapping*

 \$("#gantt").ejGantt({
baselineEndDateMapping: 'baselineEndDate'
}); | **Property:** *taskFields.baselineEndDate*

 var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
baselineEndDate: 'baselineEndDate'
}
});
ganttObj.appendTo('#gantt');|

| To map milestone mapping from data source | **Property:** *milestoneMapping*

 \$("#gantt").ejGantt({
milestoneMapping: 'isMilestone'
}); | **Property:** *taskFields.milestone*

 var ganttObj = new ej.gantt.Gantt({
taskFields: {
 milestone:
 'isMilestone'
}
});
ganttObj.appendTo('#gantt');|

| To map notes from data source | **Property:** *notesMapping*

 \$("#gantt").ejGantt({
notesMapping: 'notes'
}); | **Property:** *taskFields.notes*

 var ganttObj = new ej.gantt.Gantt({
taskFields: {
 notes:
 'notes'
}
});
ganttObj.appendTo('#gantt');|

| To map parent task id from data source | **Property:** *parentTaskIdMapping*

 \$("#gantt").ejGantt({
parentTaskIdMapping: 'parentId'
}); | **Property:** *taskFields.parentID*

 var ganttObj = new ej.gantt.Gantt({
taskFields: {
 parentID:
 'parentId'
}
});
ganttObj.appendTo('#gantt');|

| To map assigned resources from data source | **Property:** *resourceInfoMapping*

 \$("#gantt").ejGantt({
resourceInfoMapping: 'assignedResource'
}); | **Property:** *taskFields.resourceInfo*

 var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
 resourceInfo:
 'assignedResource'
}
});
ganttObj.appendTo('#gantt');|

| To map expand state from data source | **Property:** *expandStateMapping*

 \$("#gantt").ejGantt({
expandStateMapping: 'isExpanded'
}); | **Property:** *taskFields.expandState*

 var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
 expandState: 'isExpanded'
}
});
ganttObj.appendTo('#gantt');|

Members

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| To define fields in add dialog | **Property:** *addDialogFields*

 \$("#gantt").ejGantt({
{
addDialogFields: [{ field: 'taskName', editType: 'stringedit' }]

}); | **Property:** *addDialogFields*

 var ganttObj = new
 ej.gantt.Gantt({
addDialogFields: [{ type: 'General', fields: ['taskName']
 }]
});
ganttObj.appendTo('#gantt');|

| To enable/disable column resize | **Property:** *allowColumnResize*

 \$("#gantt").ejGantt({
{
allowColumnResize: true
}); | **Property:** *allowResizing*

 var ganttObj = new ej.gantt.Gantt({
allowResizing:
 true
});
ganttObj.appendTo('#gantt');|

| To enable/disable row drag and drop | **Property:** *allowDragAndDrop*

 \$(" #gantt").ejGantt(
{
allowDragAndDrop: true
}); | **Property:** *allowRowDragAndDrop*

 var ganttObj = new ej.gantt.Gantt({
allowRowDragAndDrop:
 true
}
});
ganttObj.appendTo('#gantt');|

| To enable/disable taskbar editing | **Property:** *allowGanttChartEditing*

 \$(" #gantt").ejGantt(
{
allowGanttChartEditing: true
}); | **Property:**
editSettings.allowTaskbarEditing
var ganttObj = new ej.gantt.Gantt({
editSettings:
 {
allowTaskbarEditing: true
}
});
ganttObj.appendTo('#gantt');|

| To enable/disable key navigation | **Property:** *allowKeyboardNavigation*

 \$(" #gantt").ejGantt(
{
allowKeyboardNavigation: true
}); | **Property:** *allowKeyboard*

 var ganttObj = new ej.gantt.Gantt({
allowKeyboard:
 true
});
ganttObj.appendTo('#gantt');|

| To enable/disable multiple sorting option | **Property:** *allowMultiSorting*

 \$(" #gantt").ejGantt(
{
allowMultiSorting: true
}); | **Property:** *allowSorting*

var ganttObj = new ej.gantt.Gantt({
allowSorting:
 true
});
ganttObj.appendTo('#gantt');|

| To enable/disable multiple exporting option | **Property:** *allowMultipleExporting*

 \$(" #gantt").ejGantt(
{
allowMultipleExporting: true
}); | Not applicable |

| To map baseline end date from data source | **Property:** *baselineEndDateMapping*

 \$(" #gantt").ejGantt(
{
baselineEndDateMapping: 'baselineEndDate'
}); | **Property:**
taskFields.baselineEndDate

var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
baselineEndDate: 'baselineEndDate'
}
});
ganttObj.appendTo('#gantt');|

| To map baseline start date from data source | **Property:** *baselineStartDateMapping*

 \$(" #gantt").ejGantt(
{
baselineStartDateMapping: 'baselineStartDate'
}); | **Property:**
taskFields.baselineStartDate

var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
baselineStartDate: 'baselineStartDate'
}
});
ganttObj.appendTo('#gantt');|

| To define tooltip for all cells | **Property:** *cellTooltipTemplate*

 \$(" #gantt").ejGantt(
{
cellTooltipTemplate: '#templateId'
}); | Not applicable |

| To map child task from data source | **Property:** *childMapping*

 \$(" #gantt").ejGantt(
{
childMapping : 'subTasks'
}); | **Property:** *taskFields.child*

var ganttObj = new ej.gantt.Gantt({
taskFields: {
child:
 'subTasks'
}
});
ganttObj.appendTo('#gantt');|

| To define columns fields in column insert dialog | **Property:** *columnDialogFields*

 \$(" #gantt").ejGantt(
{
columnDialogFields: ['field', 'headerText']
}); | Not applicable |

| To define working time range of day | **Property:** *dayWorkingTime*

 \$(" #gantt").ejGantt(
{
dayWorkingTime: [{from: '09:00 AM', to: '06:00 PM' }]
}); |
Property: *dayWorkingTime*

var ganttObj = new ej.gantt.Gantt({
dayWorkingTime:
 [from: 9, to: 18]
});
ganttObj.appendTo('#gantt');|

| To define tooltip template for row drag action | **Property:** *dragTooltip*

 \$(" #gantt").ejGantt(
{
dragTooltip: { showTooltip: true }
}); | Not applicable |

| To map duration from data source | **Property:** *durationMapping*

 \$("#gantt").ejGantt(

durationMapping : 'duration'
); | **Property:** *taskFields.duration*

var ganttObj = new ej.gantt.Gantt({
taskFields: {
duration:
 'duration'
}
});
ganttObj.appendTo('#gantt');|

| To map duration unit from data source | **Property:** *durationUnitMapping*

 \$("#gantt").ejGantt(

durationUnitMapping : 'durationUnit'
); | **Property:**
taskFields.durationUnit

var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
durationUnit: 'durationUnit'
}
});
ganttObj.appendTo('#gantt');|

| To define fields in edit dialog | **Property:** *editDialogFields*

 \$("#gantt").ejGantt(

editDialogFields: [{ field: 'taskName', editType: 'stringedit' }]

); | **Property:** *editDialogFields*

 var ganttObj = new
 ej.gantt.Gantt({
editDialogFields: [{ type: 'General', fields: ['TaskName']
 }]
});
ganttObj.appendTo('#gantt');|

| To define click or double click action to begin edit action | **Property:** *editSettings.allowIndent*

 \$("#gantt").ejGantt(

editSettings: { allowIndent: true
}
); | Not
 applicable|

| To enable indent/ outdent option | **Property:** *editSettings.beginEditAction*

 \$("#gantt").ejGantt(

editSettings: { beginEditAction: ej.Gantt.BeginEditAction.Click

}
); | Not applicable|

| To define edit mode in Gantt | **Property:** *editSettings.editMode*

 \$("#gantt").ejGantt(

editSettings: { editMode: normal
}
); | **Property:**
editSettings.mode

 var ganttObj = new ej.gantt.Gantt({
editSettings: { mode:
 'Normal' }
});
ganttObj.appendTo('#gantt');|

| To define new row position in Gantt | **Property:** *editSettings.rowPosition*

 \$("#gantt").ejGantt(

editSettings: { rowPosition:
 ej.Gantt.RowPosition.AboveSelectedRow
}
); | **Property:** *editSettings.newRowPosition*

 var ganttObj = new ej.gantt.Gantt({
editSettings: { newRowPosition: 'Below'
 }
});
ganttObj.appendTo('#gantt');|

| To render parent in collapsed state | **Property:** *enableCollapseAll*

 \$("#gantt").ejGantt(

enableCollapseAll: true
); | **Property:** *collapseAllParentTasks*

 var ganttObj = new ej.gantt.Gantt({
collapseAllParentTasks:
 true
});
ganttObj.appendTo('#gantt'); |

| To enable context menu in Gantt | **Property:** *enableContextMenu*

 \$("#gantt").ejGantt(

enableContextMenu: true
); | Not applicable|

| To enable progressbar resizing | **Property:** *enableProgressBarResizing*

 \$("#gantt").ejGantt(

enableProgressBarResizing: true
); | **Property:**
editSettings.allowTaskbarEditing

 var ganttObj = new ej.gantt.Gantt({
editSettings: {
 allowTaskbarEditing: true }
});
ganttObj.appendTo('#gantt'); |

| To enable serial number support | **Property:** *enableSerialNumber*

 \$("#gantt").ejGantt(

enableSerialNumber: true
); | Not applicable |

| To enable taskbar editing tooltip | **Property:** *enableTaskbarDragTooltip*

 \$("#gantt").ejGantt(
{
enableTaskbarDragTooltip: true
}); | Not Applicable|

| To enable taskbar tooltip | **Property:** *enableTaskbarTooltip*

 \$("#gantt").ejGantt(
{
enableTaskbarTooltip: true
}); | **Property:**
tooltipSettings.showTooltip

 var ganttObj = new ej.gantt.Gantt({
tooltipSettings: {
 showTooltip: true }
});
ganttObj.appendTo('#gantt'); |

| To enable virtual rendering in Gantt | **Property:** *enableVirtualization*

 \$("#gantt").ejGantt(
{
enableVirtualization : true
}); | Not Applicable|

| To enable work break down structure in Gantt | **Property:** *enableWBS*

 \$("#gantt").ejGantt(
{
enableWBS : true
}); | Not Applicable|

| To enable work break down structure predecessor in Gantt | **Property:** *enableWBSPredecessor*

 \$("#gantt").ejGantt(
{
enableWBSPredecessor : true
}); | Not Applicable|

| To map end date from data source | **Property:** *endDateMapping*

 \$("#gantt").ejGantt(
{
endDateMapping: 'endDate'
}); | **Property:** *taskFields.endDate*

var ganttObj = new ej.gantt.Gantt({
taskFields: {
endDate:
 'endDate'
}
});
ganttObj.appendTo('#gantt');|

| To map expand state from data source | **Property:** *expandStateMapping*

 \$("#gantt").ejGantt(
{
expandStateMapping: 'isExpanded'
}); | **Property:**
taskFields.expandState

var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
expandState: 'isExpanded'
}
});
ganttObj.appendTo('#gantt');|

| To define group collection for resource view Gantt | **Property:** *groupCollection*

 \$("#gantt").ejGantt(
{
groupCollection : []
}); | Not Applicable|

| To map group id for resource view Gantt | **Property:** *groupIdMapping*

 \$("#gantt").ejGantt(
{
groupIdMapping : 'groupId'
}); | Not Applicable|

| To map group name for resource view Gantt | **Property:** *groupNameMapping*

 \$("#gantt").ejGantt(
{
groupNameMapping : 'groupName'
}); | Not Applicable|

| To highlight non working time range in Gantt | **Property:** *highlightNonWorkingTime*

 \$("#gantt").ejGantt(
{
highlightNonWorkingTime : true
}); | Not Applicable|

| To define days in holiday collection | **Property:** *holidays.day*

 \$("#gantt").ejGantt(
{
holidays: [{day: '3/20/2018'}]
}); | **Property:** *holidays.from*

var ganttObj = new ej.gantt.Gantt({
holidays: [{from:
 '3/20/2018'}]
});
ganttObj.appendTo('#gantt');|

| To define left task label | **Property:** *leftTaskLabelMapping*

 \$("#gantt").ejGantt(
{
leftTaskLabelMapping: 'endDate'
}); | **Property:**
labelSettings.leftLabel

var ganttObj = new ej.gantt.Gantt({
labelSettings:
 {
leftLabel: 'endDate'
}
});
ganttObj.appendTo('#gantt');|

| To define left task label | **Property:** *leftTaskLabelTemplate*

 \$("#gantt").ejGantt(
{
leftTaskLabelMapping: '#leftLabel'
}); | **Property:**
labelSettings.leftLabel

var ganttObj = new ej.gantt.Gantt({
labelSettings:
 {
leftLabel: '#leftLabel'
}
});
ganttObj.appendTo('#gantt');|

| To map milestone mapping from data source | **Property:** *milestoneMapping*

 \$(" #gantt").ejGantt(
{
milestoneMapping: 'isMilestone'
}); | **Property:**
taskFields.milestone

var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
milestone: 'isMilestone'
}
});
ganttObj.appendTo('#gantt');|

| To define non-working background color | **Property:** *nonWorkingBackground*

 \$(" #gantt").ejGantt(
{
nonWorkingBackground : '#0000FF'
}); | Not Applicable|

| To map notes from data source | **Property:** *notesMapping*

 \$(" #gantt").ejGantt(
{
notesMapping: 'notes'
}); | **Property:** *taskFields.notes*

var ganttObj = new ej.gantt.Gantt({
taskFields: {
notes:
 'notes'
}
});
ganttObj.appendTo('#gantt');|

| To define background color for parent progress bar | **Property:** *parentProgressbarBackground*

 \$(" #gantt").ejGantt(
{
parentProgressbarBackground: '#565673'
}); | Not
 applicable|

| To map parent task id from data source | **Property:** *parentTaskIdMapping*

 \$(" #gantt").ejGantt(
{
parentTaskIdMapping: 'parentId'
}); | **Property:**
taskFields.parentID

var ganttObj = new ej.gantt.Gantt({
taskFields: {
parentID:
 'parentId'
}
});
ganttObj.appendTo('#gantt');|

| To define background color for parent taskbar | **Property:** *parentTaskbarBackground*

 \$(" #gantt").ejGantt(
{
parentTaskbarBackground: '#565673'
}); | Not applicable|

| To map predecessors from data source | **Property:** *predecessorMapping*

 \$(" #gantt").ejGantt(
{
predecessorMapping: 'predecessor'
}); | **Property:**
taskFields.dependency

var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
dependency: 'predecessor'
}
});
ganttObj.appendTo('#gantt');|

| To define tooltip template for connector line | **Property:** *predecessorTooltipTemplate*

 \$(" #gantt").ejGantt(
{
predecessorTooltipTemplate: '#predecessorTooltip'
}); |
Property: *tooltipSettings.connectorLine*

 var ganttObj = new
 ej.gantt.Gantt({
tooltipSettings: { connectorLine: '#predecessorTooltip'
 }
});
ganttObj.appendTo('#gantt'); |

| To map progress from data source | **Property:** *progressMapping*

 \$(" #gantt").ejGantt(
{
progressMapping: 'progress'
}); | **Property:** *taskFields.progress*

var ganttObj = new ej.gantt.Gantt({
taskFields: {
progress:
 'progress'
}
});
ganttObj.appendTo('#gantt');|

| To define background color for progress bar | **Property:** *progressbarBackground*

 \$(" #gantt").ejGantt(
{
progressbarBackground : '#0000FF'
}); | Not Applicable|

| To define height for progress bar | **Property:** *progressbarHeight*

 \$(" #gantt").ejGantt(
{
progressbarHeight : 80
}); | Not Applicable|

| To define template for progress resize tooltip | **Property:** *progressbarTooltipTemplate*

 \$(" #gantt").ejGantt(
{
progressbarTooltipTemplate: ""
}); | **Property:**
tooltipSettings.editing

 var ganttObj = new ej.gantt.Gantt({
tooltipSettings: { editing:
 ""
});
ganttObj.appendTo('#gantt'); |

| To define template id for progress resize tooltip | **Property:** *progressbarTooltipTemplateId*

 \$("#gantt").ejGantt(
{
progressbarTooltipTemplateId: '#progressResize'
}); |
Property: *tooltipSettings.editing*

 var ganttObj = new ej.gantt.Gantt({
tooltipSettings:
 { editing: '#progressResize' }
});
ganttObj.appendTo('#gantt'); |

| To make Gantt as read only | **Property:** *readOnly*

 \$("#gantt").ejGantt(
{
readOnly: true
}); | Not Applicable |

| To define mapping property for resource collection in resource view Gantt | **Property:**
resourceCollectionMapping

 \$("#gantt").ejGantt(
{
resourceCollectionMapping:
 'resources'
}); | Not Applicable |

| To map resource id field from resource collection | **Property:** *resourceIdMapping*

 \$("#gantt").ejGantt(
{
resourceIdMapping: 'resourceId'
}); | **Property:** *resourceFields.id*

var ganttObj = new ej.gantt.Gantt({
resourceFields: {
 id:
 'ResourceID'
}
});
ganttObj.appendTo('#gantt'); |

| To map assigned resources from data source | **Property:** *resourceInfoMapping*

 \$("#gantt").ejGantt(
{
resourceInfoMapping: 'assignedResource'
}); | **Property:**
taskFields.resourceInfo

var ganttObj = new ej.gantt.Gantt({
taskFields:
 {
resourceInfo: 'assignedResource'
}
});
ganttObj.appendTo('#gantt'); |

| To map resource unit field from assigned resource collection | **Property:** *resourceUnitMapping*

 \$("#gantt").ejGantt(
{
resourceUnitMapping: 'resourceUnit'
}); | **Property:**
resourceFields.unit

var ganttObj = new ej.gantt.Gantt({
resourceFields:
 {
 unit: 'Unit'
}
});
ganttObj.appendTo('#gantt'); |

| To define right task label | **Property:** *rightTaskLabelMapping*

 \$("#gantt").ejGantt(
{
rightTaskLabelMapping: 'endDate'
}); | **Property:**
labelSettings.rightLabel

var ganttObj = new ej.gantt.Gantt({
labelSettings:
 {
rightLabel: 'endDate'
}
});
ganttObj.appendTo('#gantt'); |

| To enable rounding off date value in taskbar editing | **Property:** *roundOffDayworkingTime*

 \$("#gantt").ejGantt(
{
roundOffDayworkingTime: true
}); | Not applicable |

| To define project end date in Gantt | **Property:** *scheduleEndDate*

 \$("#gantt").ejGantt(
{
scheduleEndDate: '3/20/2018'
}); | **Property:** *projectEndDate*

var ganttObj = new ej.gantt.Gantt({
projectEndDate:
 '3/20/2018'
});
ganttObj.appendTo('#gantt'); |

| To configure timeline settings in Gantt | **Property:** *scheduleHeaderSettings*

 \$("#gantt").ejGantt(
{
scheduleHeaderSettings:
{
weekHeaderFormat: 'MMM
 dd , yyyy',
 dayHeaderFormat: "",
yearHeaderFormat: 'yyyy',

 monthHeaderFormat: 'MMM',
hourHeaderFormat: 'HH',
 scheduleHeaderType:
 'week',
 minutesPerInterval: 'auto',
weekendBackground: "",

 timescaleStartDateMode: 'auto',
 timescaleUnitSize:'100%',
 weekStartDay: 0,

 updateTimescaleView: true
 }
}); | **Property:** *timelineSettings*

var ganttObj =
 new ej.gantt.Gantt({
timelineSettings: {
timelineViewMode:
 'Week',
timelineUnitSize: 33,
weekStartDay:
 0,
weekendBackground:"",
showTooltip: true,
updateTimescaleView:

```
true,<br/>topTier: {<br/>unit: 'Week',<br/>format: 'MMM dd, y',<br/>count: 1,<br/>formatter:
null<br/>},<br/>bottomTier: {<br/>unit: 'Day',<br/>format: 'dd',<br/>count: 1,<br/>formatter:
null<br/>}<br/>}<br/>};<br/>gantObj.appendTo('#gantt');|
```

| To define project start date in Gantt | **Property:** *scheduleStartDate*

\$("#gantt").ejGantt(
{
scheduleStartDate: '3/20/2018'
}); | **Property:**
projectStartDate

var gantObj = new ej.gantt.Gantt({
projectStartDate:
 '3/20/2018'
});
gantObj.appendTo('#gantt');|

| To define selected cell index in Gantt | **Property:** *selectedCellIndexes*

 \$("#gantt").ejGantt(
{
selectedCellIndexes: []
}); | Not applicable |

| To define selection type in Gantt | **Property:** *selectionType*

 \$("#gantt").ejGantt(
{
selectionType: ej.Gantt.SelectionType.Single
}); | **Property:**
selectionSettings.type

var gantObj = new ej.gantt.Gantt({
selectionSettings: { type:
 'Single'
});
gantObj.appendTo('#gantt');|

| To define selection mode in Gantt | **Property:** *selectionMode*

 \$("#gantt").ejGantt(
{
selectionMode: ej.Gantt.SelectionMode.Row
}); | **Property:**
selectionSettings.mode

var gantObj = new ej.gantt.Gantt({
selectionSettings: {
 mode: 'Row'
});
gantObj.appendTo('#gantt');|

| To enable/disable column chooser | **Property:** *showColumnChooser*

 \$("#gantt").ejGantt(
{
showColumnChooser: true
}); | **Property:** *showColumnMenu*

var gantObj = new ej.gantt.Gantt({
showColumnMenu:
 true
});
gantObj.appendTo('#gantt');|

| To enable/disable column add, remove option in column menu | **Property:** *showColumnOptions*

\$("#gantt").ejGantt(
{
showColumnOptions: true
}); | Not applicable |

| To enable/disable tooltip for grid cells | **Property:** *showGridCellTooltip*

 \$("#gantt").ejGantt(
{
showGridCellTooltip: true
}); | Not applicable |

| To render progress status taskbar | **Property:** *showProgressStatus*

 \$("#gantt").ejGantt(
{
showProgressStatus: true
}); | **Property:** *labelSettings.taskLabel*

var gantObj = new ej.gantt.Gantt({
labelSettings: {
taskLabel:
 '\${progress}%'
}
});
gantObj.appendTo('#gantt');|

| To render resource names right to taskbar | **Property:** *showResourceNames*

 \$("#gantt").ejGantt(
{
showResourceNames: true
}); | **Property:**
labelSettings.rightLabel

var gantObj = new ej.gantt.Gantt({
labelSettings:
 {
rightLabel: 'resourceInfo'
}
});
gantObj.appendTo('#gantt');|

| To render task name left to taskbar | **Property:** *showTaskNames*

 \$("#gantt").ejGantt(
{
showTaskNames: true
}); | **Property:** *labelSettings.leftLabel*

var gantObj = new ej.gantt.Gantt({
labelSettings: {
leftLabel:
 'taskName'
}
});
gantObj.appendTo('#gantt');|

| To define height for Gantt | **Property:** *sizeSettings.height*

 \$("#gantt").ejGantt(
{
sizeSettings: {height: '450px'
}); | **Property:** *height*


```

</></>var ganttObj = new ej.gantt.Gantt({<br>height:
'450px'<br>});<br>ganttObj.appendTo('#gantt');|

| To define width for Gantt | Property: sizeSettings.width <br><br>
$("#gantt").ejGantt(<br><br>sizeSettings: {width: '750px'} <br>}); | Property: width
<br><br>var ganttObj = new ej.gantt.Gantt({<br>width:
'750px'<br>});<br>ganttObj.appendTo('#gantt');|

| To map start date from data source | Property: startDateMapping <br><br>
$("#gantt").ejGantt(<br><br>startDateMapping: 'startDate' <br>}); | Property:
taskFields.startDate <br><br>var ganttObj = new ej.gantt.Gantt({<br>taskFields: {<br>startDate:
'startDate'<br>}<br>});<br>ganttObj.appendTo('#gantt');|

| To define strip lines in Gantt | Property: stripLines <br><br>
$("#gantt").ejGantt(<br><br>stripLines: [<br>{day: "02/06/2017",<br>label: "Project
Start",<br>lineStyle: "dotted",<br>lineColor: "Darkblue",<br>lineWidth: 2 }<br><br>}); |
Property: eventMarkers <br><br>var ganttObj = new ej.gantt.Gantt({<br>eventMarkers: [
<br>{day: new Date('04/09/2019'),<br>label: 'Research phase' <br>cssClass: 'e-custom-
stripline'}<br><br>});<br>ganttObj.appendTo('#gantt');|

| To map task collection from resources for resource view Gantt | Property: taskCollectionMapping
<br><br>$("#gantt").ejGantt(<br><br>taskCollectionMapping: 'tasks' <br>}); | Not applicable|

| To map id of task from data source | Property: taskIdMapping <br><br>
$("#gantt").ejGantt(<br><br>taskIdMapping: 'taskId' <br>}); | Property: taskFields.id
<br><br>var ganttObj = new ej.gantt.Gantt({<br>taskFields: {<br>id:
'taskId'<br>}<br>});<br>ganttObj.appendTo('#gantt');|

| To map name of task from data source | Property: taskNameMapping <br><br>
$("#gantt").ejGantt(<br><br>taskNameMapping: 'taskName' <br>}); | Property: taskFields.name
<br><br>var ganttObj = new ej.gantt.Gantt({<br>taskFields: {<br>name:
'taskName'<br>}<br>});<br>ganttObj.appendTo('#gantt');|

| To define task scheduling mode in Gantt | Property: taskSchedulingMode <br><br>
$("#gantt").ejGantt(<br><br>taskSchedulingMode: ej.Gantt.TaskSchedulingMode.Auto <br>}); |
Not applicable|

| To map task scheduling mode from data source | Property: taskSchedulingModeMapping <br><br>
$("#gantt").ejGantt(<br><br>taskSchedulingModeMapping: 'taskMode' <br>}); | Not applicable|

| To define task type in Gantt | Property: taskType <br><br>
$("#gantt").ejGantt(<br><br>taskType: ej.Gantt.TaskType.FixedUnit <br>}); | Not applicable|

| To define taskbar background type in Gantt | Property: taskbarBackground <br><br>
$("#gantt").ejGantt(<br><br>taskbarBackground: '#FF4453' <br>}); | Not applicable|

| To define tooltip template for taskbar edit action | Property: taskbarEditingTooltipTemplate <br>
<br>$("#gantt").ejGantt(<br><br>taskbarEditingTooltipTemplate: " <br>}); | Property:
tooltipSettings.editing <br><br>var ganttObj = new ej.gantt.Gantt({<br>tooltipSettings: { editing:
" <br>});<br>ganttObj.appendTo('#gantt');|

```

| To define tooltip template id for taskbar edit action | **Property:** *taskbarEditingTooltipTemplateId*

 `$("#gantt").ejGantt(

taskbarEditingTooltipTemplateId: '#templateId'
});` | **Property:** *tooltipSettings.editing*

 `var ganttObj = new ej.gantt.Gantt({
tooltipSettings: { editing: '#templateId' }
});
ganttObj.appendTo('#gantt');` |

| To define taskbar tooltip template string in Gantt | **Property:** *taskbarTooltipTemplate*

 `$("#gantt").ejGantt(

taskbarTooltipTemplate: "
});` | **Property:** *tooltipSettings.taskbar*

 `var ganttObj = new ej.gantt.Gantt({
tooltipSettings: { taskbar: "
});
ganttObj.appendTo('#gantt');` |

| To define taskbar tooltip template id in Gantt | **Property:** *taskbarTooltipTemplateId*

 `$("#gantt").ejGantt(

taskbarTooltipTemplateId: '#templateId'
});` | **Property:** *tooltipSettings.taskbar*

 `var ganttObj = new ej.gantt.Gantt({
tooltipSettings: { taskbar: '#templateId' }
});
ganttObj.appendTo('#gantt');` |

| To configure toolbar of Gantt | **Property:** *toolbarSettings*

 `$("#gantt").ejGantt(

toolbarSettings: { showToolbar: true, toolbarItems: [ej.Gantt.ToolbarItems.Add] }
});` | **Property:** *toolbar*

 `var ganttObj = new ej.gantt.Gantt({
toolbar: ['Add']
});
ganttObj.appendTo('#gantt');` |

| To enable predecessor validation task predecessor draw action | **Property:** *validateManualTasksOnLinking*

 `$("#gantt").ejGantt(

validateManualTasksOnLinking: true
});` | Not applicable |

| To define view type of Gantt | **Property:** *viewType*

 `$("#gantt").ejGantt(

viewType: ej.Gantt.ViewType.ProjectView
});` | **Property:** *viewType*

 `var ganttObj = new ej.gantt.Gantt({
viewType: 'ProjectView'
});
ganttObj.appendTo('#gantt');` |

| To define weekend background in Gantt | **Property:** *weekendBackground*

 `$("#gantt").ejGantt(

weekendBackground: '#FF5673'
});` | Not applicable |

| To define view type of Gantt | **Property:** *viewType*

 `$("#gantt").ejGantt(

viewType: ej.Gantt.ViewType.ProjectView
});` | **Property:** *viewType*

 `var ganttObj = new ej.gantt.Gantt({
viewType: 'ProjectView'
});
ganttObj.appendTo('#gantt');` |

| To map work value from data source | **Property:** *workMapping*

 `$("#gantt").ejGantt(

workMapping: 'estimatedHours'
});` | Not applicable |

| To define work unit for tasks | **Property:** *workUnit*

 `$("#gantt").ejGantt(

workUnit: ej.Gantt.WorkUnit.Hour
});` | Not applicable |

Sorting

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| Default | **Property:** *allowSorting*

 `$("#gantt").ejGantt({
allowSorting: true
});` | **Property:** *allowSorting*

 `var ganttObj = new ej.gantt.Gantt({
allowSorting: true
});
ganttObj.appendTo('#gantt');` |

| To enable/disable multiple sorting option | **Property:** *allowMultiSorting*

 \$("#gantt").ejGantt({
allowMultiSorting: true
}); | **Property:** *allowSorting*

var
 ganttObj = new ej.gantt.Gantt({
allowSorting:
 true
});
ganttObj.appendTo('#gantt');|

| Sort column Initially | **Property:** *sortSettings.sortedColumns*

 \$("#gantt").ejGantt({
allowSorting: true,
sortedColumns: [
 {field:
 "taskName", direction: ej.sortOrder.Descending }
]; | **Property:** *sortSettings.columns*

var ganttObj = new
 ej.gantt.Gantt({
allowSorting:true,
sortSettings:
 {
columns: [{ field: 'TaskID', direction: 'Ascending'
 }]}
});
ganttObj.appendTo('#gantt');|

| Clear the Sorted columns | **Method:** *clearSorting()*

 var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.clearSorting();
 | **Method:** *clearSorting()*

 var ganttObj = document.getElementById('gantt').ej2_instances[0]
 ganttObj.clearSorting();
 |

| Sort records in Gantt | **Method:** *sortColumn(mappingName, columnSortDirection)*

var
 ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.sortColumn("startDate","ascending");
 | **Method:**
sortColumn(columnName, direction,[isMultiSort])

var ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.sortColumn('startDate','asce
 nding');
 |

Filtering

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Filter column Initially | **Property:** *filterSettings.filteredColumns*

 \$("#gantt").ejGantt({
filteredColumns: [{
 value:
 "plan",
field: "taskName",
predicate:
 "and",
operator: "startswith"
}]
}); | **Property:** *filterSettings.columns*

var ganttObj = new ej.gantt.Gantt({
allowFiltering:
 true,
filterSettings: {
columns: [{ field: 'TaskName',
 matchCase: false, operator: 'startswith', predicate: 'and', value: 'Identify'
 }]}
});
ganttObj.appendTo('#gantt');|

| Filter records in Gantt | **Method:** *filterColumn(fieldName, filterOperator, filterValue, [predicate],
 [matchCase])*

var ganttObj =
 \$("#gantt").ejGantt("instance");
ganttObj.filterColumn("taskName", "startswith",
 "plan");
 | **Method:** *filterByColumn(fieldName, filterOperator, filterValue, [predicate],
 [matchCase],[ignoreAccent])*

var ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.filterByColumn('taskName',
 'startswith', 'plan');
|

| Filter multiple columns | **Method:** *filterContent(ejPredicate)*

 var ganttObj =
 \$("#gantt").ejGantt("instance");
var predicate = ej.Predicate("taskName",

```
ej.FilterOperators.equal, "planning", false)<br>&#160;&#160;&#160;&#160;&#160;.or("taskName",
ej.FilterOperators.equal, "plan budget", false)<br>&#160;&#160;&#160;&#160;&#160;.and("progress",
ej.FilterOperators.equal, 100, true);<br>ganttObj.filterContent(ejPredicate);<br>| Not
applicable |
```

```
| Clear filtered columns | Method: clearFilter() <br><br>var ganttObj =
$("#gantt").ejGantt("ejGantt");<br>ganttObj.clearFilter();<br>| Method: clearFiltering()
<br><br>var ganttObj =
document.getElementById('gantt').ej2_instances[0];<br>ganttObj.clearFiltering();<br>|
```

Searching

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

```
| Default | Property: toolbarSettings.toolbarItems <br><br>
$("#gantt").ejGantt({<br>&#160;&#160;toolbarSettings: {<br>&#160;&#160;showToolbar:
true,<br>&#160;&#160;toolbarItems : [ej.Gantt.ToolbarItems.Search]<br><br>}); | Property:
toolbar <br><br> var ganttObj = new ej.gantt.Gantt({<br>&#160;&#160;toolbar:
['Search']<br>});<br>ganttObj.appendTo('#gantt');|
```

```
| Search records in Gantt | Method: searchItem(key) <br><br>var ganttObj =
$("#gantt").ejGantt("ejGantt");<br>ganttObj.searchItem("plan");<br>| Method: search(key)
<br><br>var ganttObj =
document.getElementById('gantt').ej2_instances[0];<br>ganttObj.search('plan');<br>|
```

Selection

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

```
| Default | Property: allowSelection <br><br> $("#gantt").ejGantt({<br>&#160; allowSelection: true
<br>}); | Property: allowSelection <br><br>var ganttObj = new
ej.gantt.Gantt({<br>&#160;&#160;allowSelection: true <br>});<br>ganttObj.appendTo('#gantt'); |
```

```
| To define selection type in Gantt | Property: selectionType <br><br>
$("#gantt").ejGantt({<br>selectionType: ej.Gantt.SelectionType.Single <br>}); | Property:
selectionSettings.type <br><br>var ganttObj = new
ej.gantt.Gantt({<br>&#160;&#160;selectionSettings: { type:
'Single'<br>});<br>ganttObj.appendTo('#gantt');|
```

```
| To define selection mode in Gantt | Property: selectionMode <br><br>
$("#gantt").ejGantt({<br>selectionMode: ej.Gantt.SelectionMode.Row <br>}); | Property:
selectionSettings.mode <br><br>var ganttObj = new
ej.gantt.Gantt({<br>&#160;&#160;selectionSettings: { mode:
'Row'<br>});<br>ganttObj.appendTo('#gantt');|
```

```
| Select Row by Index | Property: selectedRowIndex <br><br> $("#gantt").ejGantt({<br>&#160;
selectedRowIndex : 2 <br>}); | Property: selectedRowIndex <br><br>var ganttObj = new
ej.gantt.Gantt({<br>&#160;&#160;selectedRowIndex : 2 <br>});<br>ganttObj.appendTo('#gantt');
|
```

| To define selected cell index in Gantt | **Property:** *selectedCellIndexes*

`$("#gantt").ejGantt({selectedCellIndexes: []});` | Not applicable |

| Select Multiple Cells | **Method:** *selectCells(Indexes,preservePreviousSelectedCell)*

 `var`

`ganttObj = $("#gantt").data("ejGantt");`
`var indexes = [{rowIndex:4, cellIndex: 4},`
`{rowIndex: 3, cellIndex: 3}];`
`ganttObj.selectCells(indexes, true);` | **Method:**
`selectCells(rowCellIndexes)`

`var ganttObj =`
`document.getElementById('gantt').ej2_instances[0];`
`var indexes = [{rowIndex:4, cellIndex:`
`4}, {rowIndex: 3, cellIndex: 3}];`
`ganttObj.selectionModule.selectCells(indexes);`
|

| Select multiple Rows | **Method:** *selectMultipleRows(rowIndexes)*

`var ganttObj =`

`$("#gantt").data("ejGantt");`
`ganttObj.selectMultipleRows([1,2,3]);`
 | **Method:**
`selectRows(key)`

`var ganttObj =`
`document.getElementById('gantt').ej2_instances[0];`
`ganttObj.selectionModule.selectRows`
`([1,2,3]);`
|

| Triggers after cell selection action | **Event:** *cellSelected*

`$("#gantt").ejGantt({`
`
cellSelected: function (args) {}
});` | **Event:** *cellSelected*

`var ganttObj= new`
`ej.gantt.Gantt({
cellSelected: function (args) {}
});`
|

| Triggers on cell selection action | **Event:** *cellSelecting*

`$("#gantt").ejGantt({`
`
cellSelecting: function (args) {}
});` | **Event:** *cellSelecting*

`var ganttObj= new`
`ej.gantt.Gantt({
cellSelecting: function (args) {}
});`
|

| Triggers after row selection action | **Event:** *rowSelected*

`$("#gantt").ejGantt({`
`
rowSelected: function (args) {}
});` | **Event:** *rowSelected*

`var ganttObj= new`
`ej.gantt.Gantt({
rowSelected: function (args) {}
});`
|

| Triggers before row selection action | **Event:** *rowSelecting*

`$("#gantt").ejGantt({`
`
rowSelecting: function (args) {}
});` | **Event:** *rowSelecting*

`var ganttObj= new`
`ej.gantt.Gantt({
rowSelecting: function (args) {}
});`
|

Editing

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----| -----| -----|

| Default | **Property:** *editSettings*

 `$("#gantt").ejGantt({
 editSettings :`
`{
 allowEditing: true,
 allowAdding:`
`true,
 allowDeleting: true,
 showDeleteConfirmDialog:`
`true}
});` | **Property:** *editSettings*

 `var ganttObj = new`
`ej.gantt.Gantt({
 editSettings: {
 allowEditing:`
`true,
 allowAdding: true,
 allowDeleting:`
`true,
 showDeleteConfirmDialog: true}
});`
`ganttObj.appendTo('#gantt');`|

| Cell Editing | **Property:** *editSettings.editMode*

`$("#gantt").ejGantt({
editSettings: {
editMode: "cellEditing"
}
});` | **Property:**
`editSettings.mode`

 `var ganttObj = new ej.gantt.Gantt({
 editSettings: {`
`mode: 'Auto' }
});`
`ganttObj.appendTo('#gantt');`|

| Dialog Editing | **Property:** *editSettings.editMode*

 \$("#gantt").ejGantt({
editSettings: {
editMode: "normal"
}
}); | **Property:** *editSettings.mode*

 var ganttObj = new ej.gantt.Gantt({
 editSettings: { mode: 'Dialog' }
});
ganttObj.appendTo('#gantt');|

| To enable/disable taskbar editing | **Property:** *allowGanttChartEditing*

 \$("#gantt").ejGantt({
allowGanttChartEditing: true
}); | **Property:** *editSettings.allowTaskbarEditing*
var ganttObj = new ej.gantt.Gantt({
editSettings: {
 allowTaskbarEditing: true
}
});
ganttObj.appendTo('#gantt');|

| To enable progressbar resizing | **Property:** *enableProgressBarResizing*

 \$("#gantt").ejGantt({
enableProgressBarResizing: true
}); | **Property:** *editSettings.allowTaskbarEditing*

 var ganttObj = new ej.gantt.Gantt({
editSettings: {
 allowTaskbarEditing: true
}
});
ganttObj.appendTo('#gantt');|

| To enable indent/ outdent option | **Property:** *editSettings.allowIndent*

 \$("#gantt").ejGantt({
editSettings: {
allowIndent: true
}
}); | Not applicable |

| To define click or double click action to begin edit action | **Property:** *editSettings.beginEditAction*

 \$("#gantt").ejGantt({
editSettings: {
beginEditAction: ej.Gantt.BeginEditAction.Click
}
}); | Not applicable |

| To define new row position in Gantt | **Property:** *editSettings.rowPosition*

 \$("#gantt").ejGantt({
editSettings: {
rowPosition: ej.Gantt.RowPosition.AboveSelectedRow
}
}); | **Property:** *editSettings.newRowPosition*

 var ganttObj = new ej.gantt.Gantt({
editSettings: {
 newRowPosition: 'Below'
}
});
ganttObj.appendTo('#gantt');|

| To define fields in edit dialog | **Property:** *editDialogFields*

 \$("#gantt").ejGantt({
editDialogFields: [{ field: 'taskName', editType: 'stringedit' }]
}); | **Property:** *editDialogFields*

 var ganttObj = new ej.gantt.Gantt({
 editDialogFields: [{ type: 'General', fields: ['TaskName'] }]
});
ganttObj.appendTo('#gantt');|

| To define fields in add dialog | **Property:** *addDialogFields*

 \$("#gantt").ejGantt({
addDialogFields: [{ field: 'taskName', editType: 'stringedit' }]
}); | **Property:** *addDialogFields*

 var ganttObj = new ej.gantt.Gantt({
 addDialogFields: [{ type: 'General', fields: ['taskName'] }]
});
ganttObj.appendTo('#gantt');|

| To make Gantt as read only | **Property:** *readOnly*

 \$("#gantt").ejGantt({
readOnly: true
}); | Not Applicable |

| To open Edit dialog | **Method:** *openEditDialog()*

var ganttObj = \$("#gantt").data("ejGantt");
ganttObj.openEditDialog();
 | **Method:** *openEditDialog()*

var ganttObj = document.getElementById('gantt').ej2_instances[0];
ganttObj.openEditDialog();
|

| To open Add dialog | **Method:** *openAddDialog()*

var ganttObj = \$("#gantt").data("ejGantt");
ganttObj.openAddDialog();
 | **Method:** *openAddDialog()*

```

<br><br>var ganttObj =
document.getElementById('gantt').ej2_instances[0];<br>ganttObj.openAddDialog();<br>|
Add task in Gantt | Method: addRecord(data, rowPosition) <br><br>var ganttObj =
$("#gantt").data("ejGantt");<br>var data =
{<br>&#160;&#160;taskId:"40",<br>&#160;&#160;taskName:"New Task
40",<br>&#160;&#160;startDate:"2/20/2014",<br>&#160;&#160;endDate:"2/25/2014"<br>};<br>
ganttObj.addRecord(data, ej.Gantt.AddRowPosition.Child);<br>| Method: addRecord(data,
rowPosition, rowIndex) <br><br>var ganttObj =
document.getElementById('gantt').ej2_instances[0];<br>var data =
{<br>&#160;&#160;taskId:"40",<br>&#160;&#160;taskName:"New Task
40",<br>&#160;&#160;startDate:"2/20/2014",<br>&#160;&#160;endDate:"2/25/2014"<br>};<br>
ganttObj.addRecord(data, 'Below', 10);<br>|
Delete selected item | Method: deleteItem() <br><br>var ganttObj =
$("#gantt").data("ejGantt");<br>ganttObj.deleteItem();<br>| Method: deleteRecord()
<br><br>var ganttObj =
document.getElementById('gantt').ej2_instances[0];<br>ganttObj.editModule.deleteRecord();<br>|
Update task details by id | Method: updateRecordByTaskId(data) <br><br>var ganttObj =
$("#gantt").data("ejGantt");<br>var data = { taskId: 4, taskName: "updated
value"};<br>ganttObj.updateRecordByTaskId(data);<br>| Method: updateRecordById
<br><br>var ganttObj = document.getElementById('gantt').ej2_instances[0];<br>var data = {
taskId: 4, taskName: "updated value"};<br>ganttObj.updateRecordById(data);<br>|
Delete dependency | Method: deleteDependency(fromTaskId,toTaskId) <br><br>var ganttObj =
$("#gantt").data("ejGantt");<br>ganttObj.deleteDependency(3, 6);<br>| Not applicable |
Save Edit | Method: saveEdit() <br><br>var ganttObj =
$("#gantt").data("ejGantt");<br>ganttObj.saveEdit();| Not applicable |
Cancel Edit | Method: cancelEdit() <br><br>var ganttObj =
$("#gantt").data("ejGantt");<br>ganttObj.cancelEdit();| Method: cancelEdit() <br><br>var
ganttObj = document.getElementById('gantt').ej2_instances[0] <br>ganttObj.cancelEdit()
Triggers for every Gantt action before it get started | Event: actionBegin
<br><br>$("#gantt").ejGantt({ <br>actionBegin: function (args) {}<br>});| Event: actionBegin
<br><br>var ganttObj= new ej.gantt.Gantt({<br>actionBegin: function (args) {}<br>});<br>|
Triggers for after every Gantt action completed | Event: actionComplete
<br><br>$("#gantt").ejGantt({ <br>actionComplete: function (args) {}<br>});| Event:
actionComplete <br><br>var ganttObj= new ej.gantt.Gantt({<br>actionComplete: function
(args) {}<br>});<br>|
Triggers while resizing, dragging the taskbar | Event: taskbarEditing <br><br>$("#gantt").ejGantt({
<br>taskbarEditing: function (args) {}<br>});| Event: taskbarEditing <br><br>var ganttObj= new
ej.gantt.Gantt({<br>taskbarEditing: function (args) {}<br>});<br>|

```



```
| Triggers after taskbar resize, drag action | Event: taskbarEdited <br/><br/>$("#gantt").ejGantt({
<br/>taskbarEdited: function (args) {}<br/>}); | Event: taskbarEdited <br/><br/>var ganttObj= new
ej.gantt.Gantt({<br/>taskbarEdited: function (args) {}<br/>});<br/>|
```

Columns

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

```
| To enable/disable column resize | Property: allowColumnResize <br/> <br/>
$("${#gantt").ejGantt({<br/>allowColumnResize: true <br/>}); | Property: allowResizing <br/><br/> var
ganttObj = new ej.gantt.Gantt({<br/>&#160;&#160;allowResizing:
true<br/>});<br/>ganttObj.appendTo('#gantt'); |
```

```
| To enable/disable column chooser | Property: showColumnChooser <br><br>
$(“#gantt”).ejGantt({<br>showColumnChooser: true <br>}); | Property: showColumnMenu
<br><br>var ganttObj = new ej.gantt.Gantt({<br>##160;##160;showColumnMenu:
true<br>});<br>gantObj.appendTo(“#gantt”);|
```

<p> To enable/disable column add, remove option in column menu Property: <i>showColumnOptions</i></p> <p>

 \$(“#gantt”).ejGantt({
showColumnOptions: true
}); Not applicable </p>
--

```
| Tree column index | Property: treeColumnIndex <br/> <br/>
$("${ganantt").ejGantt({<br/>treeColumnIndex: 2, <br/>}); | Property: treeColumnIndex <br/> <br/> var
gananttObj = new ej.gantt.Gantt({<br/>treeColumnIndex:
2<br/>});<br/>gananttObj.appendTo('#ganantt'); |
```

To define column fields in column menu	Property: <code>columnDialogFields</code>
<code>\$("#gantt").ejGantt({</code>	<code>columnDialogFields: [</code>
<code>"field", "headerText", "editType", "width",</code>	<code>"visible", "allowSorting", "textAlign", "headerTextAlign"</code>
<code>];</code>	<code>];</code>
Not applicable	

```
| Show column | Method: showColumn(headerText) <br><br> var ganttObj =  
$("#gantt").data("ejGantt");<br>ganttObj.showColumn("Task Name"); | Method:  
showColumn(keys, showBy) <br><br> var ganttObj =  
document.getElementById('gantt').ej2_instances[0] <br>ganttObj.showColumn("TaskName");|
```

```
| Hide column | Method: hideColumn(headerText) <br><br> var ganttObj =  
$(“#gantt”).data(“ejGantt”);<br>ganttObj.hideColumn(“Task Name”); | Method: hideColumn(keys,  
showBy) <br><br> var ganttObj = document.getElementById(‘gantt’).ej2_instances[0] <br>  
ganttObj.hideColumn(“TaskName”); |
```

Toolbar

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

```
| To configure default toolbars of Gantt | Property: toolbarSettings.toolbarItems <br/> <br/>
$("#gantt").ejGantt({<br/>toolbarSettings: {<br/>#160;#160;showToolBar:
true,<br/>#160;#160;toolbarItems:
[<br/>#160;#160;ej.Gantt.ToolbarItems.Add,<br/>#160;#160;ej.Gantt.ToolbarItems.Edit,<br/>
#160;#160;ej.Gantt.ToolbarItems.Delete,<br/>#160;#160;ej.Gantt.ToolbarItems.Update,<br/>
#160;#160;ej.Gantt.ToolbarItems.Cancel,<br/>#160;#160;ej.Gantt.ToolbarItems.ExpandAll,<
```



```

    <br>&#160;&#160;ej.Gantt.ToolbarItems.CollapseAll,<br>&#160;&#160;ej.Gantt.ToolbarItems.Sear
    ch,<br>&#160;&#160;ej.Gantt.ToolbarItems.PrevTimeSpan,<br>&#160;&#160;ej.Gantt.ToolbarIte
    ms.NextTimeSpan<br>],<br><br>}; | Property: toolbar <br><br> var ganttObj = new
    ej.gantt.Gantt({<br>toolbar:
    ['Add','Edit','Delete','Update','Cancel',<br>&#160;&#160;'ExpandAll','CollapseAll','Search','PrevT
    imeSpan','NextTimeSpan'],<br><br>};<br>ganttObj.appendTo('#gantt'); |

    | Other toolbars | Property: toolbarSettings.toolbarItems <br><br>
    $('#gantt').ejGantt({<br>toolbarSettings: {<br>&#160;&#160;showToolBar:
    true,<br>&#160;&#160;toolbarItems:
    [<br>&#160;&#160;ej.Gantt.ToolbarItems.Indent,<br>&#160;&#160;ej.Gantt.ToolbarItems.Outden
    t,<br>&#160;&#160;ej.Gantt.ToolbarItems.CriticalPath,<br>&#160;&#160;ej.Gantt.ToolbarItems.E
    xcelExport,<br>&#160;&#160;ej.Gantt.ToolbarItems.PdfExport<br>],<br><br>}; | Not applicable |

    | Custom toolbar | Property: toolbarSettings.customToolBarItems <br><br>
    $('#gantt').ejGantt({<br>toolbarSettings: {<br>&#160;&#160;showToolBar:
    true,<br>&#160;&#160;customToolBarItems: [{ text: "ShowBaseline", tooltipText: "Show
    Baseline" }, { text: "Reset",tooltipText:"Reset" }]<br>,<br>}; | Property: toolbar <br><br> var
    ganttObj = new ej.gantt.Gantt({<br>&#160;&#160;toolbar: [{text: 'Quick Filter', tooltipText:
    'Quick Filter', id: 'toolbarfilter', align:'Right'}],<br><br>};<br>ganttObj.appendTo('#gantt'); |

    | Triggers when toolbar items clicked | Event: toolbarClick <br><br>$('#gantt').ejGantt({
    <br>toolbarClick: function (args) {}<br>}); | Event: toolbarClick <br><br>let gantt:Gantt= new
    Gantt({<br>toolbarClick: function (args) {}<br>});<br>|
  
```

ToolTip

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

```

    | To enable taskbar tooltip | Property: enableTaskbarTooltip <br><br>
    $('#gantt').ejGantt({<br>enableTaskbarTooltip: true <br>}); | Property:
    tooltipSettings.showTooltip <br><br> var ganttObj = new
    ej.gantt.Gantt({<br>&#160;&#160;tooltipSettings: { showTooltip: true
    }<br>});<br>ganttObj.appendTo('#gantt'); |

    | To define tooltip for all cells | Property: cellTooltipTemplate <br><br>
    $('#gantt').ejGantt({<br>cellTooltipTemplate: '#templated' <br>}); | Not applicable |

    | To define tooltip template for row drag action | Property: dragTooltip <br><br>
    $('#gantt').ejGantt({<br>dragTooltip: { showTooltip: true } <br>}); | Not applicable |

    | To enable taskbar editing tooltip | Property: enableTaskbarDragTooltip <br><br>
    $('#gantt').ejGantt({<br>enableTaskbarDragTooltip: true <br>}); | Not Applicable |

    | To enable/disable tooltip for grid cells | Property: showGridCellTooltip <br><br>
    $('#gantt').ejGantt({<br>showGridCellTooltip: true <br>}); | Not applicable |

    | To enable/disable tooltip for grid cells | Property: showGridExpandCellTooltip <br><br>
    $('#gantt').ejGantt({<br>showGridExpandCellTooltip: true <br>}); | Not applicable |
  
```

| To define taskbar tooltip template in Gantt | **Property:** *taskbarTooltipTemplate*

 \$(" #gantt").ejGantt({
taskbarTooltipTemplate: 'template tooltip string'
}); | **Property:**
tooltipSettings.taskbar

 var ganttObj = new
 ej.gantt.Gantt({
 tooltipSettings: { taskbar: 'template tooltip string'
 }
});
ganttObj.appendTo('#gantt'); |

| To define taskbar tooltip template id in Gantt | **Property:** *taskbarTooltipTemplateId*

 \$(" #gantt").ejGantt({
taskbarTooltipTemplateId: '#templateId'
}); | **Property:**
tooltipSettings.taskbar

 var ganttObj = new
 ej.gantt.Gantt({
 tooltipSettings: { taskbar: '#templateId'
 }
});
ganttObj.appendTo('#gantt'); |

| To define tooltip template for connector line | **Property:** *predecessorTooltipTemplate*

 \$(" #gantt").ejGantt({
predecessorTooltipTemplate: '#predecessorTooltip'
}); | **Property:**
tooltipSettings.connectorLine

 var ganttObj = new
 ej.gantt.Gantt({
 tooltipSettings: { connectorLine: '#predecessorTooltip'
 }
});
ganttObj.appendTo('#gantt'); |

| To define template for progress resize tooltip | **Property:** *progressbarTooltipTemplate*

 \$(" #gantt").ejGantt({
progressbarTooltipTemplate: 'template tooltip string'
}); |
Property: *tooltipSettings.editing*

 var ganttObj = new
 ej.gantt.Gantt({
 tooltipSettings: { editing: 'template tooltip string'
 }
});
ganttObj.appendTo('#gantt'); |

| To define template id for progress resize tooltip | **Property:** *progressbarTooltipTemplateId*

 \$(" #gantt").ejGantt({
progressbarTooltipTemplateId: '#progressResize'
}); | **Property:**
tooltipSettings.editing

 var ganttObj = new
 ej.gantt.Gantt({
 tooltipSettings: { editing: '#progressResize'
 }
});
ganttObj.appendTo('#gantt'); |

| To define tooltip template for taskbar edit action | **Property:** *taskbarEditingTooltipTemplate*

 \$(" #gantt").ejGantt({
taskbarEditingTooltipTemplate: 'template tooltip string'
}); |
Property: *tooltipSettings.editing*

 var ganttObj = new
 ej.gantt.Gantt({
 tooltipSettings: { editing: 'template tooltip string'
 }
});
ganttObj.appendTo('#gantt'); |

| To define tooltip template id for taskbar edit action | **Property:** *taskbarEditingTooltipTemplateId*

 \$(" #gantt").ejGantt({
taskbarEditingTooltipTemplateId: '#templateId'
}); |
Property: *tooltipSettings.editing*

 var ganttObj = new
 ej.gantt.Gantt({
 tooltipSettings: { editing: '#templateId'
 }
});
ganttObj.appendTo('#gantt'); |

Timeline

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| To configure timeline settings in Gantt | **Property:** *scheduleHeaderSettings*

 \$(" #gantt").ejGantt({
scheduleHeaderSettings:
{
weekHeaderFormat: 'MMM dd ,
 yyyy',
dayHeaderFormat: "dd,MM,yy",
yearHeaderFormat: 'yyyy',


```
monthHeaderFormat: 'MMM',<br/>hourHeaderFormat: 'HH',<br/> scheduleHeaderType:
'week', <br/> minutesPerInterval: 'auto',<br/>weekendBackground: '#F2F2F2',<br/>
timescaleStartDateMode: 'auto', <br/> timescaleUnitSize:'100%', <br/> weekStartDay: 0,<br/>
updateTimescaleView: true <br/> } <br/>}); | Property: timelineSettings <br/><br/>var ganttObj =
new ej.gantt.Gantt({<br/>timelineSettings: {<br/>&#160;timelineViewMode:
'Week',<br/>&#160;timelineUnitSize: 33,<br/>&#160;weekStartDay: 0,<br/>&#160;showTooltip:
true,<br/>&#160;weekendBackground: ' ',<br/>&#160;updateTimescaleView:
true,<br/>&#160;topTier: {<br/>&#160;&#160;&#160;unit:
'Week',<br/>&#160;&#160;&#160;format: 'MMM dd, y',<br/>&#160;&#160;&#160;count:
1,<br/>&#160;&#160;&#160;formatter: null<br/>},<br/>&#160;bottomTier:
{<br/>&#160;&#160;&#160;unit: 'Day',<br/>&#160;&#160;&#160;format:
'dd',<br/>&#160;&#160;&#160;count: 1,<br/>&#160;&#160;&#160;formatter:
null<br/>}<br/>}<br/>});<br/>ganttObj.appendTo('#gantt');|
```

| To define weekend background in Gantt | **Property:** weekendBackground

 \$("#gantt").ejGantt({
weekendBackground: '#FF5673'
}); | Not applicable |

| To Highlight weekends | **Property:** highlightWeekends

 \$("#gantt").ejGantt({

highlightWeekends: true
}); | **Property:** highlightWeekends

var ganttObj = new
ej.gantt.Gantt({
 highlightWeekends : true

});
ganttObj.appendTo('#gantt'); |

| To include weekends | **Property:** includeWeekend

 \$("#gantt").ejGantt({

includeWeekend : true
}); | **Property:** includeWeekend

var ganttObj = new
ej.gantt.Gantt({
 includeWeekend : true

});
ganttObj.appendTo('#gantt'); |

| To define project start date in Gantt | **Property:** scheduleStartDate

\$("#gantt").ejGantt({
scheduleStartDate: '3/20/2018'
}); | **Property:**
 projectStartDate

var ganttObj = new
 ej.gantt.Gantt({
 projectStartDate:
 '3/20/2018'
});
ganttObj.appendTo('#gantt'); |

| To define project end date in Gantt | **Property:** scheduleEndDate

 \$("#gantt").ejGantt({
scheduleEndDate: '3/20/2018'
}); | **Property:** projectEndDate

var ganttObj = new ej.gantt.Gantt({
 projectEndDate:
 '3/20/2018'
});
ganttObj.appendTo('#gantt'); |

| Update project start date and end date | **Method:** updateScheduleDates(startDate,endDate)

var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.updateScheduleDates("5/25/2017",
 "9/27/2017");
 | **Method:** updateProjectDates(startDate, endDate, isTimelineRoundOff)

var ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.updateProjectDates("5/25/2017",
 "9/27/2017", true);
|

Rows

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| To enable/disable row drag and drop | **Property:** *allowDragAndDrop*

 \$("#gantt").ejGantt({
allowDragAndDrop: true
}); | **Property:** *allowRowDragAndDrop*

var ganttObj = new ej.gantt.Gantt({
 allowRowDragAndDrop:
 true
});
ganttObj.appendTo('#gantt'); |

| To enable/disable alternate row background | **Property:** *enableAltRow*

 \$("#gantt").ejGantt({
enableAltRow: true
}); | Not applicable |

| To add Row height | **Property:** *rowHeight*

 \$("#gantt").ejGantt({
 rowHeight: 60

}); | **Property:** *rowHeight*

var ganttObj = new
 ej.gantt.Gantt({
 rowHeight : 60
});
ganttObj.appendTo('#gantt'); |

| To render parent in collapsed state | **Property:** *enableCollapseAll*

 \$("#gantt").ejGantt({
enableCollapseAll: true
}); | **Property:** *collapseAllParentTasks*

 var ganttObj = new ej.gantt.Gantt({
 collapseAllParentTasks:
 true
});
ganttObj.appendTo('#gantt'); |

| Expand/collapse record by id | **Method:** *expandCollapseRecord(taskId)*

var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.expandCollapseRecord(1);
 | **Method:**
collapseById() *expandById()*

var ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.expandById(1);

gan
 ttObj.collapseById(1);
|

| Expand all rows | **Method:** *expandAllItems()*

var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.expandAllItems();
 | **Method:** *expandAll()*

var ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.expandAll();
|

| Collapse all rows | **Method:** *collapseAllItems()*

var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.collapseAllItems();
 | **Method:** *collapseAll()*

var ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.collapseAll();
|

| Triggers after row collapse action | **Event:** *collapsed*

\$("#gantt").ejGantt({
collapsed:
 function (args) {}
}); | **Event:** *collapsed*

var ganttObj= new
 ej.gantt.Gantt({
collapsed: function (args) {}
});
|

| Triggers before row collapse action | **Event:** *collapsing*

\$("#gantt").ejGantt({

collapsing: function (args) {}
}); | **Event:** *collapsing*

var ganttObj= new
 ej.gantt.Gantt({
collapsing: function (args) {}
});
|

| Triggers after Gantt row was expanded | **Event:** *expanded*

\$("#gantt").ejGantt({

expanded: function (args) {}
}); | **Event:** *expanded*

var ganttObj= new
 ej.gantt.Gantt({
expanded: function (args) {}
});
|

| Triggers before Gantt row expand action | **Event:** *expanding*

\$("#gantt").ejGantt({

expanding: function (args) {}
}); | **Event:** *expanding*

var ganttObj= new
 ej.gantt.Gantt({
expanding: function (args) {}
});
|

| Triggers before grid rows render action | **Event:** *rowDataBound*

\$(“#gantt”).ejGantt({
rowDataBound: function (args) {}
}); | **Event:** *rowDataBound*

var ganttObj= new ej.gantt.Gantt({
rowDataBound: function (args) {}
});
 |

| Triggers while dragging a row | **Event:** *rowDrag*

\$(“#gantt”).ejGantt({
rowDrag: function (args) {}
}); | **Event:** *rowDrag*

var ganttObj= new ej.gantt.Gantt({
rowDrag: function (args) {}
});
 |

| Triggers while while start to drag row | **Event:** *rowDragStart*

\$(“#gantt”).ejGantt({
rowDragStart: function (args) {}
}); | **Event:** *rowDragStart*

var ganttObj= new ej.gantt.Gantt({
rowDragStart: function (args) {}
});
 |

| Triggers while while drop a row | **Event:** *rowDragStop*

\$(“#gantt”).ejGantt({
rowDragStop: function (args) {}
}); | **Event:** *rowDrop*

var ganttObj= new ej.gantt.Gantt({
rowDrop: function (args) {}
});
 |

Resources

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| ----- | ----- | ----- |

| To map resources | **Property:** *resources*

\$(“#gantt”).ejGantt({
 resources: projectResources,
}); | **Property:** *resources*

var ganttObj = new ej.gantt.Gantt({
 resources: projectResources
});
ganttObj.appendTo(“#gantt”); |

| To map resource id field from resource collection | **Property:** *resourceIdMapping*

\$(“#gantt”).ejGantt({
resourceIdMapping: ‘resourceId’
}); | **Property:** *resourceFields.id*

var ganttObj = new ej.gantt.Gantt({
resourceFields: {
 id: ‘ResourceID’

});
ganttObj.appendTo(“#gantt”); |

| To map resource name field from resource collection | **Property:** *resourceNameMapping*

\$(“#gantt”).ejGantt({
resourceNameMapping: ‘resourceName’
}); | **Property:** *resourceFields.name*

var ganttObj = new ej.gantt.Gantt({
resourceFields: {
 name: ‘ResourceName’

});
ganttObj.appendTo(“#gantt”); |

| To map resource unit field from assigned resource collection | **Property:** *resourceUnitMapping*

\$(“#gantt”).ejGantt({
resourceUnitMapping: ‘resourceUnit’
}); | **Property:** *resourceFields.unit*

var ganttObj = new ej.gantt.Gantt({
resourceFields: {
 unit: ‘ResourceUnit’

});
ganttObj.appendTo(“#gantt”); |

| To define resource view type of Gantt | **Property:** *viewType*

\$(“#gantt”).ejGantt({
viewType: ej.Gantt.ViewType.ResourceView
}); | **Property:** *viewType*

var ganttObj = new ej.gantt.Gantt({
 viewType: ‘ResourceView’
});
ganttObj.appendTo(“#gantt”); |

| To define mapping property for resource collection in resource view Gantt | **Property:** *resourceCollectionMapping*

\$(“#gantt”).ejGantt({
resourceCollectionMapping: ‘resources’
}); | Not Applicable |

| To map task collection from resources for resource view Gantt | **Property:** *taskCollectionMapping*

\$(“#gantt”).ejGantt({
taskCollectionMapping: ‘tasks’
}); | Not applicable |

| To map group id for resource view Gantt | **Property:** *groupIdMapping*

 \$("#gantt").ejGantt({
groupIdMapping : 'groupId'
}); | Not Applicable|

| To map group name for resource view Gantt | **Property:** *groupNameMapping*

 \$("#gantt").ejGantt({
groupNameMapping : 'groupName'
}); | Not Applicable|

Baseline

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To render baseline | **Property:** *renderBaseline*

 \$("#gantt").ejGantt({

renderBaseline: true
}); | **Property:** *renderBaseline*

var ganttObj = new
ej.gantt.Gantt({
 renderBaseline: true
});
ganttObj.appendTo('#gantt'); |

| To define baselineColor | **Property:** *baselineColor*

 \$("#gantt").ejGantt({

baselineColor: "#fba41c"
}); | **Property:** *baselineColor*

var ganttObj = new
ej.gantt.Gantt({
 baselineColor: 'red'
});
ganttObj.appendTo('#gantt'); |

Context Menu

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To enable context menu | **Property:** *enableContextMenu*

 \$("#gantt").ejGantt({

enableContextMenu: true
}); | **Property:** *enableContextMenu*

var ganttObj = new
ej.gantt.Gantt({
 enableContextMenu: true

});
ganttObj.appendTo('#gantt'); |

| To define custom menu items | **Event:** *contextMenuOpen*

\$("#gantt").ejGantt({

contextMenuOpen: function(args)
{
 args.contextMenuItems.push({
 headerText:
"Expand/Collapse",
 menuId:
"expand",
 iconPath: "url(Expand-02-
WF.png)",
 eventHandler: function()
{
 event handler for custom menu
items
 }
});
}); | **Property:** *contextMenuItems*

var
ganttObj = new ej.gantt.Gantt({
contextMenuItems: [
 { text: 'Collapse the Row',
target: '.e-content', id: 'collapserow' },
 { text: 'Expand the Row', target: '.e-content',
id: 'expandrow' }
];
});
ganttObj.appendTo('#gantt'); |

| Triggers before context menu opens | **Event:** *contextMenuOpen*

\$("#gantt").ejGantt({

contextMenuOpen: function (args) {}
}); | **Event:** *contextMenuOpen*

var
ganttObj= new ej.gantt.Gantt({
contextMenuOpen: function (args) {}
});
|

Scheduling Tasks

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To define task scheduling mode in Gantt | **Property:** *taskSchedulingMode*

 \$("#gantt").ejGantt({
taskSchedulingMode: ej.Gantt.TaskSchedulingMode.Auto
}); |

Property: *taskMode*

var ganttObj = new ej.gantt.Gantt({ taskMode: Auto
});
ganttObj.appendTo('#gantt');|

| To map task scheduling mode from data source | **Property:** *taskSchedulingModeMapping*

\$("#gantt").ejGantt({
taskSchedulingModeMapping: 'taskMode'
}); | **Property:** *taskFields.manual*

var ganttObj = new ej.gantt.Gantt({
taskFields: {
 manual: 'isManual'
}
});
ganttObj.appendTo('#gantt');|

| To enable schedule date validation while task predecessor draw action | **Property:** *validateManualTasksOnLinking*

\$("#gantt").ejGantt({
validateManualTasksOnLinking: true
}); | **Property:** *validateManualTasksOnLinking*

var ganttObj = new ej.gantt.Gantt({
 validateManualTasksOnLinking: true
});
ganttObj.appendTo('#gantt');|

| To define working time range of day | **Property:** *dayWorkingTime*

\$("#gantt").ejGantt({
dayWorkingTime: [{from: '09:00 AM', to: '06:00 PM' }]
}); | **Property:** *dayWorkingTime*

var ganttObj = new ej.gantt.Gantt({
 dayWorkingTime: [from: 9, to: 18]
});
ganttObj.appendTo('#gantt'); |

| To enable rounding off date value in taskbar editing | **Property:** *roundOffDayworkingTime*

\$("#gantt").ejGantt({
roundOffDayworkingTime: true
}); | Not applicable |

| To define non-working background color | **Property:** *nonWorkingBackground*

\$("#gantt").ejGantt({
nonWorkingBackground : '#0000FF'
}); | Not Applicable |

| To highlight non working time range in Gantt | **Property:** *highlightNonWorkingTime*

\$("#gantt").ejGantt({
highlightNonWorkingTime : true
}); | Not Applicable |

To set working days of a week | **Property:** *workweek*

\$("#gantt").ejGantt({
workweek:["Sunday","Monday","Tuesday","Wednesday","Thursday"]
}); | **Property:** *workWeek*

var ganttObj = new ej.gantt.Gantt({
 workWeek: ["Sunday","Monday","Tuesday","Wednesday","Thursday"]
});
ganttObj.appendTo('#gantt'); |

| To enable/disable Unscheduled tasks | **Property:** *allowUnscheduledTask*

\$("#gantt").ejGantt({
 allowUnscheduledTask: true
}); | **Property:** *allowUnscheduledTasks*

var ganttObj = new ej.gantt.Gantt({
 allowUnscheduledTasks: true
});
ganttObj.appendTo('#gantt'); |

Appearance and Customizations

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

| To define taskbar background type in Gantt | **Property:** *taskbarBackground*

\$("#gantt").ejGantt({
taskbarBackground: '#FF4453'
}); | Not applicable |

| To define background color for parent taskbar | **Property:** *parentTaskbarBackground*

 \$(" #gantt").ejGantt({
parentTaskbarBackground: '#565673'
}); | Not applicable |

| To add Taskbar height | **Property:** *taskbarHeight*

 \$(" #gantt").ejGantt({

 taskbarHeight: 30,
 rowHeight: 40
}); | **Property:** *taskbarHeight*

var
 ganttObj = new ej.gantt.Gantt({
 taskbarHeight:50,
 rowHeight
 : 60
});
ganttObj.appendTo('#gantt'); |

| To define background color for parent progress bar | **Property:** *parentProgressbarBackground*

 \$(" #gantt").ejGantt({
parentProgressbarBackground: '#565673'
}); | Not
 applicable |

| To define background color fro progress bar | **Property:** *progressbarBackground*

 \$(" #gantt").ejGantt({
progressbarBackground : '#0000FF'
}); | Not Applicable |

| To define height for progress bar | **Property:** *progressbarHeight*

 \$(" #gantt").ejGantt({
progressbarHeight : 80
}); | Not Applicable |

| To render progress status taskbar | **Property:** *showProgressStatus*

 \$(" #gantt").ejGantt({
showProgressStatus: true
}); | **Property:** *labelSettings.taskLabel*

var ganttObj = new ej.gantt.Gantt({
 labelSettings:
 {
 taskLabel: '\$ {progress}%'
}
});
ganttObj.appendTo('#gantt'); |

| To set connectorline width | **Property:** *connectorlineWidth*

 \$(" #gantt").ejGantt({
 connectorLineWidth: 2
}); | **Property:** *connectorLineWidth*

var ganttObj = new ej.gantt.Gantt({
 connectorLineWidth: 2

});
ganttObj.appendTo('#gantt'); |

| To set connectorline background | **Property:** *connectorLineBackground*

 \$(" #gantt").ejGantt({
 connectorLineBackground: "#0aecb8"
}); | **Property:**
connectorLineBackground

var ganttObj = new
 ej.gantt.Gantt({
 connectorLineBackground:"red"

});
ganttObj.appendTo('#gantt'); |

| To define weekend background in Gantt | **Property:** *weekendBackground*

 \$(" #gantt").ejGantt({
weekendBackground: '#FF5673'
}); | Not applicable

| To define taskbar template | **Property:** *taskbarTemplate*

 \$(" #gantt").ejGantt({

 taskbarTemplate: "#taskbarTemplate",
}); | **Property:** *taskbarTemplate*

var ganttObj =
 new ej.gantt.Gantt({
 taskbarTemplate: '#TaskbarTemplate',

});
ganttObj.appendTo('#gantt'); |

| To define parent taskbar template | **Property:** *parentTaskbarTemplate*

 \$(" #gantt").ejGantt({
 parentTaskbarTemplate: "#parentTaskbarTemplate",
}); |
Property: *parentTaskbarTemplate*

var ganttObj = new
 ej.gantt.Gantt({
 parentTaskbarTemplate: '#ParentTaskbarTemplate',

});
ganttObj.appendTo('#gantt'); |

| To define milestone template | **Property:** *milestoneTemplate*

 \$(" #gantt").ejGantt({
 milestoneTemplate: "#milestoneTemplate",
}); | **Property:**
milestoneTemplate

var ganttObj = new


```

ej.gantt.Gantt({<br>&#160;&#160;milestoneTemplate: '#MilestoneTemplate',
<br>});<br>gantObj.appendTo('#gant'); |

| To define right task label | Property: rightTaskLabelMapping <br><br>
$("#gant").ejGantt({<br>rightTaskLabelMapping: 'endDate' <br>}); | Property:
labelSettings.rightLabel <br><br>var gantObj = new ej.gantt.Gantt({<br>&#160;labelSettings:
{<br>&#160;&#160;rightLabel: 'endDate'<br><br>});<br>gantObj.appendTo('#gant');|

| To define left task label | Property: leftTaskLabelMapping <br><br>
$("#gant").ejGantt({<br>leftTaskLabelMapping: 'endDate' <br>}); | Property:
labelSettings.leftLabel <br><br>var gantObj = new ej.gantt.Gantt({<br>&#160;labelSettings:
{<br>&#160;&#160;leftLabel: 'endDate'<br><br>});<br>gantObj.appendTo('#gant');|

| To define right task label template | Property: rightTaskLabelTemplate <br><br>
$("#gant").ejGantt({<br>rightTaskLabelTemplate: "#rightLabelTemplate", <br>}); | Property:
labelSettings.rightLabel <br><br>var gantObj = new ej.gantt.Gantt({<br>&#160;labelSettings:
{<br>&#160;&#160;rightLabel: '#rightLabel'<br><br>});<br>gantObj.appendTo('#gant');|

| To define left task label template | Property: leftTaskLabelTemplate <br><br>
$("#gant").ejGantt({<br>leftTaskLabelTemplate: "#leftLabelTemplate", <br>}); | Property:
labelSettings.leftLabel <br><br>var gantObj = new ej.gantt.Gantt({<br>&#160;labelSettings:
{<br>&#160;&#160;leftLabel: '#leftLabel'<br><br>});<br>gantObj.appendTo('#gant');|

| To render resource names right to taskbar | Property: showResourceNames <br><br>
$("#gant").ejGantt({<br>showResourceNames: true <br>}); | Property: labelSettings.rightLabel
<br><br>var gantObj = new ej.gantt.Gantt({<br>&#160;labelSettings:
{<br>&#160;&#160;rightLabel: 'resourceInfo'<br><br>});<br>gantObj.appendTo('#gant');|

| To render task name left to taskbar | Property: showTaskNames <br><br>
$("#gant").ejGantt({<br>showTaskNames: true <br>}); | Property: labelSettings.leftLabel
<br><br>var gantObj = new ej.gantt.Gantt({<br>&#160;labelSettings:
{<br>&#160;&#160;leftLabel: 'taskName'<br><br>});<br>gantObj.appendTo('#gant');|

| Triggers on taskbar rendering action | Event: queryTaskbarInfo <br><br>$("#gant").ejGantt({
<br>queryTaskbarInfo: function (args) {}<br>}); | Event: queryTaskbarInfo <br><br>var
gantObj= new ej.gantt.Gantt({<br>queryTaskbarInfo: function (args) {}<br>});<br>|

| Triggers on grid cell rendering action | Event: queryCellInfo <br><br>$("#gant").ejGantt({
<br>queryCellInfo: function (args) {}<br>}); | Event: queryCellInfo <br><br>var gantObj= new
ej.gantt.Gantt({<br>queryCellInfo: function (args) {}<br>});<br>|

```

Stripline

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----| -----| -----|

```

| To define striplines | Property: stripLines <br><br>
$("#gant").ejGantt({<br>stripLines:
[<br>&#160;&#160;day: "01/02/2014",<br>&#160;&#160;label: "Project
Release",<br>&#160;&#160;lineStyle: "dotted",<br>&#160;&#160;lineColor:
"blue",<br>&#160;&#160;lineWidth: 2<br>]}<br>}); | Property: eventMarkers <br><br>var gantObj =
new ej.gantt.Gantt({<br>eventMarkers:[<br>&#160;&#160;day:

```

```
'04/10/2019',<br>&#160;&#160;cssClass: 'e-custom-event-marker',<br>&#160;&#160;label:
'Project approval and kick-off'<br>}}<br>});<br>gantObj.appendTo('#gantt'); |
```

Holidays

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To define holidays | **Property:** *holidays*

 `$("#gantt").ejGantt({
 holidays: [{
 day: "2/03/2014",
 label: " Public holiday",
 background: "yellow"
}]
});` | **Property:** *holidays*

`var gantObj = new ej.gantt.Gantt({
 holidays: [{
 from: "04/04/2019",
 to:"04/05/2019",
 label: " Public holidays",
 cssClass:"e-custom-holiday"
}]
});
gantObj.appendTo('#gantt');` |

| To define days in holiday collection | **Property:** *holidays.day*

 `$("#gantt").ejGantt({
holidays: [{day: '3/20/2018'}]
});` | **Property:** *holidays.from*

`var gantObj = new ej.gantt.Gantt({
 holidays: [{from: '3/20/2018'}]
});
gantObj.appendTo('#gantt');` |

Others

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| To define height for Gantt | **Property:** *sizeSettings.height*

 `$("#gantt").ejGantt({
sizeSettings: {height: '450px'}
});` | **Property:** *height*

`var gantObj = new ej.gantt.Gantt({
 height: '450px'
});
gantObj.appendTo('#gantt');` |

| To define width for Gantt | **Property:** *sizeSettings.width*

 `$("#gantt").ejGantt({
sizeSettings: {width: '750px'}
});` | **Property:** *width*

`var gantObj = new ej.gantt.Gantt({
 width: '750px'
});
gantObj.appendTo('#gantt');` |

| To change splitter position | **Property:** *splitterPosition*

 `$("#gantt").ejGantt({
splitterPosition : "50%"
});` | Not applicable |

| To change splitter by position | **Property:** *splitterSettings.position*

 `$("#gantt").ejGantt({
splitterSettings: {
 position: "50%"
},
});` | **Property:** *splitterSettings.position*

 `var gantObj = new ej.gantt.Gantt({
 splitterSettings: {
 position: "50%"
}
});
gantObj.appendTo('#gantt');` |

| To change splitter by index | **Property:** *splitterSettings.index*

 `$("#gantt").ejGantt({
splitterSettings: {
 index: 3
},
});` | **Property:** *splitterSettings.columnIndex*

 `var gantObj = new ej.gantt.Gantt({
 splitterSettings: {
 columnIndex: 3
}
});
gantObj.appendTo('#gantt');` |

| To define view type of Gantt | **Property:** *viewType*

 `$("#gantt").ejGantt({
viewType: ej.Gantt.ViewType.ProjectView
});` | **Property:** *viewType*

 `var ganttObj = new ej.gantt.Gantt({
 viewType: 'ProjectView'
});
ganttObj.appendTo('#gantt');` |

| To enable Localization | **Property:** *locale*

 `$("#gantt").ejGantt({
 locale: "fr-FR"
});` | **Property:** *locale*

 `var ganttObj = new ej.gantt.Gantt({
 locale: "fr-FR"
});
ganttObj.appendTo('#gantt');` |

| To specify the date format for Gantt | **Property:** *dateFormat*

 `$("#gantt").ejGantt({
 dateFormat: "dd/MM/yyyy"
});` | **Property:** *dateFormat*

 `var ganttObj = new ej.gantt.Gantt({
 dateFormat: 'dd/MM/yyyy'
});
ganttObj.appendTo('#gantt');` |

| To enable/disable key navigation | **Property:** *allowKeyboardNavigation*

 `$("#gantt").ejGantt({
allowKeyboardNavigation: true
});` | **Property:** *allowKeyboard*

 `var ganttObj = new ej.gantt.Gantt({
 allowKeyboard: true
});
ganttObj.appendTo('#gantt');` |

| To enable serial number support | **Property:** *enableSerialNumber*

 `$("#gantt").ejGantt({
enableSerialNumber: true
});` | Not applicable |

| To enable/disable predecessor validation | **Property:** *enablePredecessorValidation*

 `$("#gantt").ejGantt({
 enablePredecessorValidation: true
});` | **Property:** *enablePredecessorValidation*

 `var ganttObj = new ej.gantt.Gantt({
 enablePredecessorValidation: true
});
ganttObj.appendTo('#gantt');` |

| To set timescale for working hours | **Property:** *workingTimeScale*

 `$("#gantt").ejGantt({
 workingTimeScale: ej.Gantt.workingTimeScale.TimeScale24Hours
});` | **Property:** *dayWorkingTime*

 `var ganttObj = new ej.gantt.Gantt({
 dayWorkingTime: [from: 0, to: 24]
});
ganttObj.appendTo('#gantt');` |

| To enable work break down structure in Gantt | **Property:** *enableWBS*

 `$("#gantt").ejGantt({
enableWBS : true
});` | Not Applicable |

| To enable work break down structure predecessor in Gantt | **Property:** *enableWBSPredecessor*

 `$("#gantt").ejGantt({
enableWBSPredecessor : true
});` | Not Applicable |

| To map work value from data source | **Property:** *workMapping*

 `$("#gantt").ejGantt({
workMapping: 'estimatedHours'
});` | **Property:** *taskFields.work*

 `var ganttObj = new ej.gantt.Gantt({
taskFields: {
 work: 'estimatedHours'
}
});
ganttObj.appendTo('#gantt');` |

| To define work unit for tasks | **Property:** *workUnit*

 `$("#gantt").ejGantt({
workUnit: ej.Gantt.WorkUnit.Hour
});` | **Property:** *workUnit*

 `var ganttObj = new ej.gantt.Gantt({
 workUnit: 'Hour'
});
ganttObj.appendTo('#gantt');` |

| To define task type in Gantt | **Property:** *taskType*

 `$("#gantt").ejGantt({
taskType: ej.Gantt.TaskType.FixedUnit
});` | **Property:** *taskType*

 `var ganttObj = new ej.gantt.Gantt({
taskType: 'FixedUnit'
});
ganttObj.appendTo('#gantt');` |

| To enable/disable multiple exporting option | **Property:** *allowMultipleExporting*

 `$("#gantt").ejGantt({
allowMultipleExporting: true
});` | Not applicable |

| To enable virtual rendering in Gantt | **Property:** *enableVirtualization*

 `$("#gantt").ejGantt({
enableVirtualization : true
});` | Not Applicable |

| Change splitter position dynamically | **Method:** *setSplitterIndex(index)* *setSplitterPosition(width)*

 `var ganttObj = $("#gantt").data("ejGantt");
ganttObj.setSplitterIndex(3);

ganttObj.ejGantt("setSplitterPosition", "40%");
 | Method: setSplitterPosition(value,type)

 var ganttObj = document.getElementById('gantt').ej2_instances[0]
 ganttObj.setSplitterPosition('40%', 'position');

 ganttObj.setSplitterPosition(3, 'columnIndex'); |`

| To destroy Gantt | **Method:** *destroy()*

 `var ganttObj = $("#gantt").data("ejGantt");
ganttObj.destroy();
 | Method: destroy()

 var ganttObj = document.getElementById('gantt').ej2_instances[0]
 ganttObj.destroy(); |`

| To update task id | **Method:** *updateTaskId(currentId, newId)*

 `var ganttObj = $("#gantt").data("ejGantt");
ganttObj.updateTaskId(5, 7);
 | Not applicable |`

| To set scroll top for Gantt | **Method:** *setScrollTop(top)*

 `var ganttObj = $("#gantt").data("ejGantt");
ganttObj.setScrollTop(50);
 | Method: setScrollTop()

 var ganttObj = document.getElementById('gantt').ej2_instances[0]
 ganttObj.setScrollTop(200); |`

| To get columns to edit in resource view | **Method:** *getResourceViewEditColumns()*

 `var ganttObj = $("#gantt").data("ejGantt");
columns = ganttObj.getResourceViewEditColumns();
 | Not applicable |`

| Show/hide critical path in Gantt | **Method:** *showCriticalPath(isShown)*

 `var ganttObj = $("#gantt").data("ejGantt");
ganttObj.showCriticalPath(true);
 | Not applicable |`

| Triggers on initialization of Gantt control | **Event:** *load*

 `$("#gantt").ejGantt({
load: function (args) {}
});` | **Event:** *load*

 `var ganttObj= new ej.gantt.Gantt({
load: function (args) {}
});
 |`

| Triggers after splitter resize action | **Event:** *splitterResized*

 `$("#gantt").ejGantt({
splitterResized: function (args) {}
});` | **Event:** *splitterResized*

 `var ganttObj= new ej.gantt.Gantt({
splitterResized: function (args) {}
});
 |`

| Triggers when taskbar item is clicked | **Event:** *taskbarClick*

 `$("#gantt").ejGantt({
taskbarClick: function (args) {}
});` | **Event:** *onTaskbarClick*

 `var ganttObj= new ej.gantt.Gantt({
onTaskbarClick: function (args) {}
});
 |`

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Add task in Gantt | **Method:** *addRecord(data, rowPosition)*
 var ganttObj =
 \$("#gantt").data("ejGantt");
var data = {
taskId:"40",
taskName:"New Task
 40",
startDate:"2/20/2014",
endDate:"2/25/2014"};
ganttObj.addRecord(data,
 ej.Gantt.AddRowPosition.Child);
 | **Method:** *addRecord(data, rowPosition, rowIndex)*
 var
 ganttObj = document.getElementById('gantt').ej2_instances[0];
var data =
 {
taskId:"40",
taskName:"New Task
 40",
startDate:"2/20/2014",
endDate:"2/25/2014"};
ganttObj.addRecord(data,
 'Below', 10);
|

| Clear filtered columns | **Method:** *clearFilter()*
 var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.clearFilter();
| **Method:** *clearFiltering()*
 var
 ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.clearFiltering();
|

| Delete selected item | **Method:** *deleteItem()*
 var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.deleteItem();
| **Method:** *deleteRecord()*
 var
 ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.editModule.deleteRecord();

 br/>|

| Expand/collapse record by id | **Method:** *expandCollapseRecord(taskId)*
 var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.expandCollapseRecord(1);
| **Method:**
collapseById()
 var ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.expandById(1);
ganttObj
 j.collapseById(1);
|

| Filter records in Gantt | **Method:** *filterColumn(fieldName, filterOperator, filterValue, [predicate],
 [matchCase])*
 var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.filterColumn("taskName", "startswith",
 "plan");
| **Method:** *filterByColumn(fieldName, filterOperator, filterValue, [predicate],
 [matchCase],[ignoreAccent])*
 var ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.filterByColumn('taskName',
 'startswith', 'plan');
|

| Search records in Gantt | **Method:** *searchItem(key)*
 var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.searchItem("plan");
| **Method:** *search(key)*

 var ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.search('plan');
|

| Select multiple rows in Gantt | **Method:** *selectMultipleRows(rowIndexes)*
 var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.selectMultipleRows([1,2,3]);
| **Method:**
selectRows(key)
 var ganttObj =
 document.getElementById('gantt').ej2_instances[0];
ganttObj.selectionModule.selectRows
 ([1,2,3]);
|

| Show/hide critical path in Gantt | **Method:** *showCriticalPath(isShown)*
 var ganttObj =
 \$("#gantt").data("ejGantt");
ganttObj.showCriticalPath(true);
| Not applicable |

| Sort records in Gantt | **Method:** *sortColumn(mappingName, columnSortDirection)*
 var ganttObj = \$("#gantt").data("ejGantt");
ganttObj.sortColumn("startDate","ascending");
|
Method: *sortColumn(columnName, direction,[isMultiSort])*
 var ganttObj = document.getElementById('gantt').ej2_instances[0];
ganttObj.sortColumn('startDate','ascending');
|

| Update task details by id | **Method:** *updateRecordByTaskId(data)*
 var ganttObj = \$("#gantt").data("ejGantt");
var data = { taskId: 4, taskName: "updated value"};
ganttObj.updateRecordByTaskId(data);
| **Method:** *updateRecordById*
 var ganttObj = document.getElementById('gantt').ej2_instances[0];
var data = { taskId: 4, taskName: "updated value"};
ganttObj.updateRecordById(data);
|

| Update project start date and end date | **Method:** *updateScheduleDates(startDate,endDate)*
 var ganttObj = \$("#gantt").data("ejGantt");
ganttObj.updateScheduleDates("5/25/2017", "9/27/2017");
| **Method:** *updateProjectDates(startDate, endDate, isTimelineRoundOff)*
 var ganttObj = document.getElementById('gantt').ej2_instances[0];
ganttObj.updateProjectDates("5/25/2017", "9/27/2017", true);
|

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Triggers for every Gantt action before it get started | **Event:** *actionBegin*
 \$("#gantt"), {
actionBegin: function (args){}
}; | **Event:** *actionBegin*
let gantt: Gantt = new Gantt({
actionBegin: function (args){}
});
|

| Triggers for after every Gantt action completed | **Event:** *actionComplete*
 \$("#gantt"), {
actionComplete: function (args){}
}; | **Event:** *actionComplete*
let gantt: Gantt = new Gantt({
actionComplete: function (args){}
});
|

| Triggers after cell selection action | **Event:** *cellSelected*
 \$("#gantt"), {
cellSelected: function (args){}
}; | **Event:** *cellSelected*
let gantt: Gantt = new Gantt({
cellSelected: function (args){}
});
|

| Triggers on cell selection action | **Event:** *cellSelecting*
 \$("#gantt"), {
cellSelecting: function (args){}
}; | **Event:** *cellSelecting*
let gantt: Gantt = new Gantt({
cellSelecting: function (args){}
});
|

| Triggers after row collapse action | **Event:** *collapsed*
 \$("#gantt"), {
collapsed: function (args){}
}; | **Event:** *collapsed*
let gantt: Gantt = new Gantt({
collapsed: function (args){}
});
|

| Triggers before row collapse action | **Event:** *collapsing*
 \$("#gantt"), {
collapsing: function (args){}
}; | **Event:** *collapsing*
let gantt: Gantt = new Gantt({
collapsing: function (args){}
});
|

|Triggers after Gantt rendered | **Event:** *create*
\$("#gantt"), {
create: function (args){
}};| **Event:** *create*
let gantt: Gantt = new Gantt({
create: function (args){
}};
|

|Triggers after Gantt row was expanded | **Event:** *expanded*
\$("#gantt"), {
expanded: function (args){
}};| **Event:** *expanded*
let gantt: Gantt = new Gantt({
expanded: function (args){
}};
|

|Triggers before Gantt row expand action | **Event:** *expanding*
\$("#gantt"), {
expanding: function (args){
}};| **Event:** *expanding*
let gantt: Gantt = new Gantt({
expanding: function (args){
}};
|

|Triggers on initialization of Gantt control | **Event:** *load*
\$("#gantt"), {
load: function (args){
}};| **Event:** *load*
let gantt: Gantt = new Gantt({
load: function (args){
}};
|

|Triggers on grid cell rendering action | **Event:** *queryCellInfo*
\$("#gantt"), {
queryCellInfo: function (args){
}};| **Event:** *queryCellInfo*
let gantt: Gantt = new Gantt({
queryCellInfo: function (args){
}};
|

|Triggers on taskbar rendering action | **Event:** *queryTaskbarInfo*
\$("#gantt"), {
queryTaskbarInfo: function (args){
}};| **Event:** *queryTaskbarInfo*
let gantt: Gantt = new Gantt({
queryTaskbarInfo: function (args){
}};
|

|Triggers before grid rows render action | **Event:** *rowDataBound*
\$("#gantt"), {
rowDataBound: function (args){
}};| **Event:** *rowDataBound*
let gantt: Gantt = new Gantt({
rowDataBound: function (args){
}};
|

|Triggers after row selection action | **Event:** *rowSelected*
\$("#gantt"), {
rowSelected: function (args){
}};| **Event:** *rowSelected*
let gantt: Gantt = new Gantt({
rowSelected: function (args){
}};
|

|Triggers before row selection action | **Event:** *rowSelecting*
\$("#gantt"), {
rowSelecting: function (args){
}};| **Event:** *rowSelecting*
let gantt: Gantt = new Gantt({
rowSelecting: function (args){
}};
|

|Triggers after splitter resize action | **Event:** *splitterResized*
\$("#gantt"), {
splitterResized: function (args){
}};| **Event:** *splitterResized*
let gantt: Gantt = new Gantt({
splitterResized: function (args){
}};
|

|Triggers after taskbar resize, drag action | **Event:** *taskbarEdited*
\$("#gantt"), {
taskbarEdited: function (args){
}};| **Event:** *taskbarEdited*
let gantt: Gantt = new Gantt({
taskbarEdited: function (args){
}};
|

|Triggers while resizing, dragging the taskbar | **Event:** *taskbarEditing*
\$("#gantt"), {
taskbarEditing: function (args){
}};| **Event:** *taskbarEditing*
let gantt: Gantt = new Gantt({
taskbarEditing: function (args){
}};
|

|Triggers when toolbar items clicked | **Event:** *toolbarClick*
\$("#gantt"), {
toolbarClick: function (args){
}};| **Event:** *toolbarClick*
let gantt: Gantt = new Gantt({
toolbarClick: function (args){
}};
|

Loading animation in EJ2 JavaScript Gantt control

The loading indicator is used to display a visual indicator while the Gantt is fetching data or performing certain actions, such as sorting or filtering. The gantt support two indicator types, which is achieved by setting the [loadingIndicator.indicatorType](#) property to Shimmer or Spinner. The default value of the indicator type is "Spinner."

In the following sample, the Shimmer indicator is displayed while the gantt is scrolled when using the virtual data.

INDEX.TS

```
import { Gantt, Sort, Selection, VirtualScroll, Filter } from
 '@syncfusion/ej2-gantt';
import { virtualData } from 'datasource.ts';
Gantt.Inject(Sort, Selection, VirtualScroll, Filter);
let gantt: Gantt = new Gantt({
  dataSource: virtualData,
  height: '450px',
  allowSorting: true,
  allowFiltering: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    parentID: 'parentID'
  },
  enableVirtualization: true,
  loadingIndicator: { indicatorType: 'Shimmer' },
  columns: [
    { field: 'TaskID' },
    { field: 'TaskName' },
    { field: 'StartDate' },
    { field: 'Duration' },
    { field: 'Progress' },
  ],
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">
  <link href="https://cdn.syncfusion.com/ej2-notifications/material.css"
rel="stylesheet" type="text/css">
```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Style and appearance in EJ2 JavaScript Gantt control

To modify the Gantt Chart appearance, you need to override the default CSS of gantt chart. Please find the list of CSS classes and its corresponding section in Gantt Chart. Also, you have an option to create your own custom theme for all the JavaScript controls using our [Theme Studio](#).

Section | CSS Class | Purpose of Class

Root | e-gantt | This class is in the root element (div) of the gantt chart control.

Header | e-gridheader | This class is added in the root element of header element. In this class, You can override thin line between header and content of the gantt chart.

| e-table | This class is added at 'table' of the gantt chart header. This CSS class makes table width as 100 %.

| e-columnheader | This class is added at 'tr' of the gantt chart header.

Grid Content | e-gridcontent | This class is added at root of body content. This is to override background color of the body.

| e-table | This class is added to table of content. This CSS class makes table width as 100 %.

| e=row | This class is added to rows of gantt chart.

| e-altrow | This class is added to alternate rows of gantt chart. This is to override alternate row color of the gantt chart.

| e-rowcell | This class is added to all cells in the gantt chart. This is to override cells appearance and styling.

Chart Content | e-gantt-chart | This class is added to the chart side of the gantt chart.

| e-chart-row | This class is added to rows of gantt chart.

Timeline | e-timeline-header-container | This class is added to timeline of the gantt chart.

| e-header-cell-label | This class is added to the header cell of the timeline.

|e-weekend-header-cell| This class is added to the weekend cells.

Taskbar|e-taskbar-main-container| This class is added to taskbar of the gantt chart.

|e-gantt-parent-taskbar| This class is added to the parent task bar of the gantt chart.

|e-gantt-milestone| This class is added to the milestone tasks of the gantt chart.

|e-gantt-unscheduled-taskbar| This class is added to the unscheduled tasks.

|e-gantt-manualparenttaskbar| This class is added to the manual scheduled parent taskbar.

|e-gantt-child-manualtaskbar| This class is added to the manual scheduled child taskbar.

|e-gantt-unscheduled-manualtask| This class is added to the manual unscheduled tasks.

Splitter|e-split-bar| This class is added to the gantt chart splitter.

|e-resize-handler| This class is added to the resize handler of the gantt chart splitter.

|e-arrow-left| This class is added to the left arrow of the resize handler.

|e-arrow=right| This class is added to the right arrow of the resize handler.

Connector Lines|e-line| This class is added to the connector lines.

|e-connector-line-right-arrow| This class is added to the right arrow of the connector line.

|e-connector-line-left-arrow| This class is added to the left arrow of the connector line.

Labels|e-task-label| This class is added to the task labels.

|e-right-label-container| This class is added to the right label.

|e-left-label-container| This class is added to the left label.

Event Markers|e-event-markers| This class is added to the event markers.

Baseline|e-baseline-bar| This class is added to the baseline.

|e-baseline-gantt-milestone-container| This class is added to the baseline of milestone tasks.

Tooltip|e-gantt-tooltip| This class is added to the tooltip.

How To

Open add edit dialog in EJ2 JavaScript Gantt control

Gantt add and edit dialogs can be opened dynamically by using [openAddDialog](#) and [openEditDialog](#) methods. The following code example shows how to open add and dialog on separate button click actions.

INDEX.TS

```
import { Gantt, Edit, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { GanttData } from 'datasource.ts';
import { getValue } from '@syncfusion/ej2-base';
Gantt.Inject(Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
```

```

        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        child: 'subtasks'
    },
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowTaskbarEditing: true
    }
});
ganttt.appendTo('#Gantt');
let editBtn: Button = new Button();
editBtn.appendTo('#editDialog');
let addBtn: Button = new Button();
addBtn.appendTo('#addDialog');
document.getElementById('editDialog').addEventListener('click', () => {
    ganttt.editModule.dialogModule.openEditDialog(getValue('TaskID',
    ganttt.selectionModule.getSelectedRecords()[0]));
});
document.getElementById('addDialog').addEventListener('click', () => {
    ganttt.editModule.dialogModule.openAddDialog();
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
    rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>
    <button id="editDialog">Open EditDialog</button> <button
    id="addDialog">Open AddDialog</button>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

NOTE: We should select any one of the row in Gantt to open the edit dialog.

Change schedule date in EJ2 JavaScript Gantt control

In the Gantt control, you can change the schedule start and end dates by clicking the custom button programmatically using the [updateProjectDates](#) method. You can pass the start and end dates as method arguments to the [updateProjectDates](#) method. You can also pass the Boolean value as an additional parameter, which is used to round-off the schedule start and end dates displayed in Gantt timeline.

INDEX.TS

```

import { Gantt, Edit, Selection } from '@syncfusion/ej2-gantt';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from 'datasource.ts';
Gantt.Inject(Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: data,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  }
});
gantt.appendTo('#Gantt');
let dateBtn: Button = new Button();
dateBtn.appendTo('#updateSchedule');
document.getElementById('updateSchedule').addEventListener('click', () => {
  gantt.updateProjectDates(new Date('01/10/2019'), new Date('06/20/2019'),
  true);
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

        <button id="updateSchedule">Change Schedule Dates</button>
        <div id="container">
            <div id="Gantt"></div>
        </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Copy paste records in EJ2 JavaScript Gantt control

You can copy and paste a record in the Gantt chart by using the `addRecord` method and `custom context menu`. It is also possible to copy and paste the parent record with multiple hierarchical child records on the required position.

INDEX.TS

```

import { Gantt, Edit, Selection, ContextMenu } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, Selection, ContextMenu);
var copiedRecord: any;
var enableFlag: any;
let gantt: Gantt = new Gantt({
    dataSource: GanttData,
    taskFields: {
        id: 'TaskID',
        name: 'TaskName',
        startDate: 'StartDate',
        duration: 'Duration',
        progress: 'Progress',
        dependency: 'Predecessor',
        child: 'subtasks'
    },
    editSettings: {
        allowAdding: true,
        allowEditing: true,
        allowDeleting: true,
        allowTaskbarEditing: true
    },
    toolbar: ['Add', 'Edit', 'Update', 'Delete', 'Cancel', 'ExpandAll',
'CollapseAll'],
    allowSelection: true,
    enableContextMenu: true,
    contextMenuItems: [
        { text: 'Copy', target: '.e-content', id: 'copy' },
        { text: 'Paste', target: '.e-content', id: 'paste' },
    ],

```

```

        contextMenuClick: function (args) {
            if (args.item.id === 'copy') {
                copiedRecord = args.rowData;
                copiedRecord.taskData.TaskID = gantt.currentViewData.length + 1;
            }
            if (args.item.id === 'paste') {
                gantt.addRecord(copiedRecord.taskData, 'Below', args.rowData.index);
                if (copiedRecord.hasChildRecords) {
                    addChildRecords(copiedRecord, args.rowData.index + 1);
                }
                copiedRecord = undefined;
            }
        },
        contextMenuOpen: function (args) {
            if (args.type !== 'Header') {
                if (copiedRecord) {
                    args.hideItems.push('Copy');
                } else {
                    args.hideItems.push('Paste');
                }
            }
        },
    });
    gantt.appendTo('#Gantt');
    function addChildRecords(record: any, index: any) {
        for (var i=0; i<record.childRecords.length; i++) {
            var childRecord = record.childRecords[i];
            childRecord.taskData.TaskID = gantt.currentViewData.length + 1;
            gantt.addRecord(childRecord.taskData, 'Child', index);
            if (childRecord.hasChildRecords) {
                addChildRecords(childRecord, index + (i+1));
            }
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```
<div id="container">
  <div id="Gantt"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Maintain record index in EJ2 JavaScript Gantt control

Row dropped record's index position can be maintained in the Gantt chart by changing the database table index position using the `rowDrop` event. In this event, the `fromIndex` and `dropIndex` values can be passed to the server side using Ajax request. On the server side, the `insert` and `insertAtTop` methods are used to update the row index position. The following code snippets explain the solution.

`ts

```
import { Gantt, Selection, Edit, RowDD, IGanttData } from '../src/index';
```

```
import { Ajax } from '@syncfusion/ej2-base';
```

```
Gantt.Inject(Selection, Edit, RowDD );
```

```
let gantt: Gantt = new Gantt({
  dataSource: new DataManager({
    url: 'https://localhost:44379/Home/UrlDatasource',
    adaptor: new UrlAdaptor,
    crossDomain: true,
    batchUrl: 'https://localhost:44379/Home/BatchUpdate'
  }),
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  allowRowDragAndDrop: true,
```

```

rowDrop: function (args) {
let record = this.flatData[args.fromIndex][this.taskFields.id];
let record2 = this.flatData[args.dropIndex][this.taskFields.id];
let data: IGanttData = args.data[0];
let uri = 'https://localhost:44379/Home/RowDropMethod';
let dragdropdata = {
record: data[0].taskData,
position: args.dropIndex,
dragidMapping: record,
dropidMapping: record2
};
let ajax = new Ajax(
{
url: uri,
type: 'POST',
contentType: "application/json",
data: JSON.stringify(dragdropdata),
});
ajax.send();
}
});
gantt.appendTo('#ganttContainer');
`ts
public IActionResult RowDropMethod([FromBody]DragDropData value)
{
var data = new CRUDModel();
copyRecord = true;
if (value.position == "bottomSegment" || value.position == "topSegment")
{
{
var childCount = 0;
int parent1 = (int)value.record.parentID;

```



```
childCount += FindChildRecords(parent1);
if (childCount == 1)
{
    var i = 0;
    for (; i < DataList.Count; i++)
    {
        if (DataList[i].taskID == parent1)
        {
            DataList[i].isParent = false;
            break;
        }
        if (DataList[i].taskID == value.record.taskID)
        {
            DataList[i].parentID = null;
            break;
        }
    }
}
DataList.Remove(DataList.Where(ds => ds.taskID == value.dragidMapping).FirstOrDefault());
var j = 0;
for (; j < DataList.Count; j++)
{
    if (DataList[j].taskID == value.dropidMapping)
    {
        value.record.parentID = DataList[j].parentID;
        break;
    }
}
data.Value = value.record;
if (value.position == "bottomSegment")
{
    this.Insert(data, value.dropidMapping);
}
```

```

}
else if (value.position == "topSegment")
{
this.InsertAtTop(data, value.dropidMapping);
}
}
else if (value.position == "middleSegment")
{
DataList.Remove(DataList.Where(ds => ds.taskID == value.dragidMapping).FirstOrDefault());
value.record.parentID = value.dropidMapping;
FindDropdata(value.dropidMapping);
data.Value = value.record;
this.Insert(data, value.dropidMapping);
}
copyRecord = false;
return Json(new { updatedRecords = value.record });
}
`

```

Custom field in EJ2 JavaScript Gantt control

Generally in Gantt, Custom fields are displayed in the Custom Tab of the Add/Edit dialogs. However, they can be included in the General Tab of Add/Edit Dialog Box using `actionBegin` and `actionComplete` events. These events are used to append the custom field to the dialog box. The following code snippets demonstrate the solution.

INDEX.TS

```

import { Gantt, Edit, Toolbar } from '@syncfusion/ej2-gantt';
import { CheckBox } from "@syncfusion/ej2-buttons";
import { TextBox, NumericTextBox, MaskedTextBox } from "@syncfusion/ej2-inputs";
import { DatePicker, DateTimePicker } from "@syncfusion/ej2-calendars";
import { DropDownList } from "@syncfusion/ej2-dropdowns";
import { GanttData } from 'datasource.ts';
Gantt.Inject(Edit, Toolbar);
var divElement;
var inputs = {
  booleanedit: CheckBox,
  dropdownedit: DropDownList,
  datepickeredit: DatePicker,
  datetimeredit: DateTimePicker,
  maskededit: MaskedTextBox,
  numericedit: NumericTextBox,
  stringedit: TextBox
}

```

```

};
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
  },
  editSettings: {
    allowAdding: true,
    allowEditing: true,
    allowDeleting: true,
    mode: "Auto"
  },
  columns: [
    { field: 'TaskID', width: '150' },
    { field: 'TaskName', width: '250' },
    { field: 'StartDate', width: '250' },
    { field: 'Duration', width: '250' },
    { field: 'Progress', width: '250' },
    { field: 'CustomField', width: '250' }
  ],
  toolbar: ['Add', 'Cancel', 'CollapseAll', 'Delete', 'Edit', 'ExpandAll',
'Update'],
  editDialogFields: [
    { type: 'General', headerText: 'General' },
    { type: 'Dependency' },
    { type: 'Resources' },
    { type: 'Notes' }
  ],
  addDialogFields: [
    { type: 'General', headerText: 'General' },
    { type: 'Dependency' },
    { type: 'Resources' },
    { type: 'Notes' }
  ],
  actionBegin: function(args) {
    if (args.requestType === "beforeOpenEditDialog" || args.requestType
=== "beforeOpenAddDialog" ) {
      var column = this.columnByField["CustomField"];
      divElement = this.createElement("div", {
        className: "e-edit-form-column"
      });
      var inputElement;
      inputElement = this.createElement("input", {
        attrs: {
          type: "text",
          id: this.controlId + "" + column.field,
          name: column.field,
          title: column.field
        }
      });
      divElement.appendChild(inputElement);
    }
  }
});

```

```

        var input = {
            enabled: true,
            floatLabelType: "Auto",
            placeholder: "CustomField",
            value: args.rowData.CustomField
        };
        var inputObj = new inputs[column.editType](input);
        inputObj.appendTo(inputElement);
    }
},
actionComplete: function(args) {
    if (args.requestType === "openEditDialog" || args.requestType ===
"openAddDialog") {
        var generalTab = document.getElementById(
            this.controlId + "GeneralTabContainer"
        );
        generalTab.appendChild(divElement);
    }
}
});
ganttt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Maintain zoom to fit in EJ2 JavaScript Gantt control

In the Gantt control, While performing edit actions or dynamically change dataSource, the timeline gets refreshed. When zoomToFit toolbar item is clicked and perform editing actions or dynamically change dataSource, the timeline gets refreshed. So that, the timeline will not fit to the project any more.

Maintain zoomToFit after edit actions

We can maintain zoomToFit after editing actions(cell edit,dialog edit,taskbar edit) by using [fitToProject](#) method in `actionComplete` and `taskbarEdited` event.

INDEX.TS

```
import { Gantt, Toolbar, Edit, Selection } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
Gantt.Inject(Toolbar, Edit, Selection);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  allowSelection: true,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  },
  toolbar: ['Edit', 'ZoomToFit'],
  editSettings: {
    allowEditing: true,
    allowTaskbarEditing: true,
  },
  labelSettings: {
    leftLabel: 'TaskName'
  },
  projectStartDate: new Date('03/24/2019'),
  projectEndDate: new Date('04/28/2019'),
  taskbarEdited: function(args) {
    if (args) {
      var obj = document.getElementById("Gantt").ej2_instances[0];
      obj.dataSource = GanttData;
      obj.fitToProject();
    }
  },
  actionComplete: function (args) {
    if ((args.action === "CellEditing" || args.action === "DialogEditing")
    && args.requestType === "save") {
      var obj = document.getElementById("Gantt").ej2_instances[0];
      obj.dataSource = GanttData;
      obj.fitToProject();
    }
  }
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Maintain zoomToFit after change dataSource dynamically

We can maintain **zoomToFit** after change **dataSource** dynamically, by calling [fitToProject](#) method in **dataBound** event.

INDEX.TS

```

import { Gantt, Toolbar } from '@syncfusion/ej2-gantt';
import { GanttData, data } from 'datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
Gantt.Inject(Toolbar);
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    dependency: 'Predecessor',
    child: 'subtasks'
  }
});

```

```

    },
    toolbar: ['ZoomToFit'],
    labelSettings: {
        leftLabel: 'TaskName'
    },
    projectStartDate: new Date('03/24/2019'),
    projectEndDate: new Date('04/28/2019'),
    dataBound: function (args) {
        this.fitToProject();
    }
});
ganttt.appendTo('#Gantt');
let changeBtn: Button = new Button();
changeBtn.appendTo('#changeData');
document.getElementById('changeData').addEventListener('click', () => {
    var obj = document.getElementById('Gantt').ej2_instances[0];
    obj.dataSource = data;
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="changeData">Change Data</button>
        <div id="Gantt"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop in EJ2 JavaScript Gantt control

In Gantt, it is possible to drag a record from another component and drop it in Gantt chart with updating the Gantt record. Here, dragging an item from **TreeView** component to Gantt and that item is updated as a resource for the Gantt record, we can achieve this, by using [nodeDragStop](#) event of **TreeView** control.

INDEX.TS

```
import { Gantt, Edit, Selection } from '@syncfusion/ej2-gantt';
import { TreeView, DragAndDropEventArgs } from '@syncfusion/ej2-
navigations';
import { editingData, editingResources } from 'datasource.ts';
import { closest } from '@syncfusion/ej2-base';
Gantt.Inject(Edit, Selection);
let treeObj: TreeView = new TreeView({
    fields: { dataSource: editingResources, id: 'resourceId', text:
'resourceName' },
    allowDragAndDrop: true,
    height: "200px",
    nodeDragStop: function (args: DragAndDropEventArgs): void {
        let ganttObj: any =
document.getElementById('Gantt').ej2_instances[0];
        let chartEle: any = closest(args.target, '.e-chart-row');
        let gridEle: any = closest(args.target, '.e-row');
        if(gridEle){
            var index = ganttObj.treeGrid.getRows().indexOf(gridEle);
            ganttObj.selectRow(index);
        }
        if (chartEle) {
            var index = chartEle.ariaRowIndex;
            ganttChart.selectRow(Number(index));
        }
        let record: any = args.draggedNodeData;
        let selectedData = ganttObj.flatData[ganttObj.selectedRowIndex];
        let selectedDataResource = selectedData.taskData.resources;
        let resources = [];
        if (selectedDataResource) {
            for (var i = 0; i < selectedDataResource.length; i++) {
                resources.push(selectedDataResource[i].resourceId);
            }
        }
        resources.push(parseInt(record.id));
        if (chartEle || gridEle) {
            var data = {
                TaskID: selectedData.taskData.TaskID,
                resources: resources
            };
            gantt.updateRecordByID(data);
            var elements = document.querySelectorAll('.e-drag-item');
            while (elements.length > 0 && elements[0].parentNode) {
                elements[0].parentNode.removeChild(elements[0]);
            }
        }
    }
});
let gantt: Gantt = new Gantt({
```



```

dataSource: editingData,
resources: editingResources,
height: '450px',
taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks',
    dependency: 'Predecessor',
    resourceInfo: 'resources'
},
resourceFields: {
    id: 'resourceId',
    name: 'resourceName'
},
labelSettings: {
    leftLabel: 'TaskName',
    rightLabel: 'resources'
},
splitterSettings: {
    position: "50%"
},
editSettings: {
    allowEditing: true
}
});
treeObj.appendTo('#TreeView');
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Gantt</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Gantt Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <b><label>Gantt</label></b>

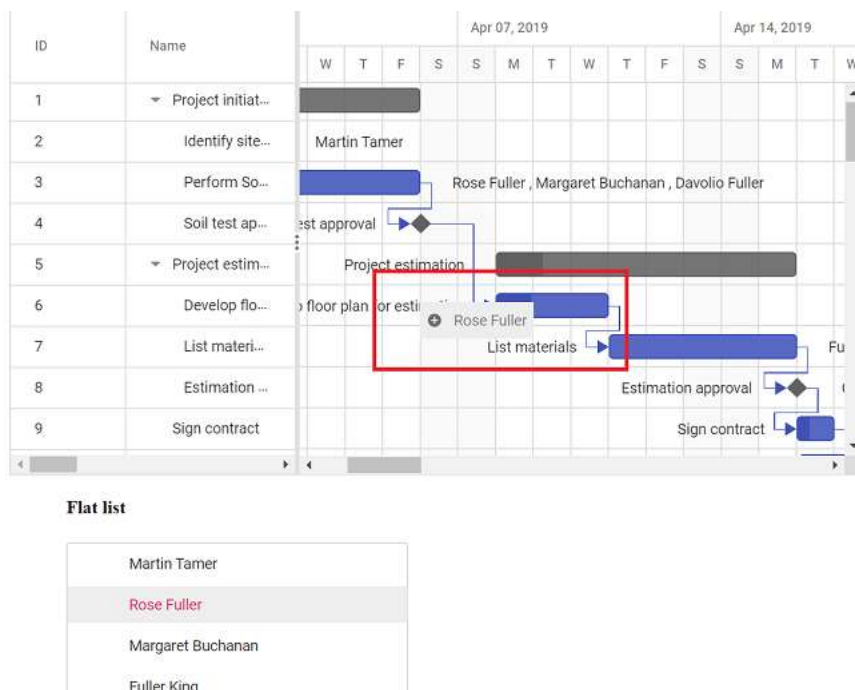
```

```

<div id="Gantt"></div>
<b><label>List</label></b>
<div id="TreeView"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following screenshot shows dropping record from another component in to Gantt, and **Rose Fuller** is added as resource for the task **Develop floor plan estimation**.



New row position in EJ2 JavaScript Gantt control

In Gantt, a new row can be added in one of the following positions: Top, Bottom, Above, Below and Child. This position can be specified through the `newRowPosition` property. We can make use of the `toolbarClick` event to create a context menu that specifies the position in which the new row is to be added when adding a record through toolbar click.

The following code snippets demonstrate how to achieve this.

INDEX.TS

```

import { ContextMenu, MenuItemModel, ContextMenuModel } from
 '@syncfusion/ej2-navigations';
import { Gantt, Toolbar, Filter, Edit, Selection } from '@syncfusion/ej2-
 gantt';
import { EmitType } from '@syncfusion/ej2-base';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { GanttData } from 'datasource.ts';

```

```

import { enableRipple } from '@syncfusion/ej2-base';
Gantt.Inject(Toolbar, Selection, Filter, Edit);
enableRipple(true);
let menuItems: MenuItemModel[] = [
    {
        text: 'Top'
    },
    {
        text: 'Bottom'
    },
    {
        text: 'Above'
    },
    {
        text: 'Below'
    },
    {
        text: 'Child'
    }
],];
let menuOptions: ContextMenuModel = {
    items: menuItems,
    select: select
};
let menuObj: ContextMenu = new ContextMenu(menuOptions, '#contextmenu');
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.text === 'Add') {
        menuObj.open(60, 20);
    }
};
function select(args: any) {
    let gantt: any = (document.getElementsByClassName('e-gantt')[0] as
any).ej2_instances[0];
    if (args.item.text === "Bottom") {
        gantt.editSettings.newRowPosition = "Bottom";
        gantt.openAddDialog();
    } else if (args.item.text === "Above") {
        if (gantt.selectedRowIndex == -1) {
            alert("Please select any row");
        } else {
            gantt.editSettings.newRowPosition = "Above";
            gantt.openAddDialog();
        }
    } else if (args.item.text === "Below") {
        if (gantt.selectedRowIndex == -1) {
            alert("Please select any row");
        } else {
            gantt.editSettings.newRowPosition = "Below";
            gantt.openAddDialog();
        }
    } else if (args.item.text === "Child") {
        if (gantt.selectedRowIndex == -1) {
            alert("Please select any row");
        } else {
            gantt.editSettings.newRowPosition = "Child";
            gantt.openAddDialog();
        }
    } else if (args.item.text === "Top") {

```

```

    gantt.editSettings.newRowPosition = "Top";
    gantt.openAddDialog();
  }
}
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  allowSelection: true,
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  editSettings: {
    allowEditing: true,
    allowAdding: true,
    allowDeleting: true
  },
  toolbarClick: clickHandler,
  toolbar: [ 'Edit', { text: 'Add', tooltipText: 'Add', id: 'Add' } ],
});
gantt.appendTo('#Gantt');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
    <ul id="contextmenu"></ul>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render timeline from 1 to 365 days in EJ2 JavaScript Gantt control

Gantt chart contains different types of in-built timeline view modes and it can be defined by using [timelineViewMode](#) property and also we can customize the timescale mode of top tier and bottom tier by using [topTier.unit](#) and [bottomTier.unit](#) properties. Timeline tier's unit can be defined by using [unit](#) property and [format](#) property was used to define date format of timeline cell and [formatter](#) property was used to define custom method to format the date value of timeline cell.

In the [bottomTier.unit](#) timescale mode, it is possible to display timeline from 1 to 365 days by making use of the formatter in the [timelineSettings](#) property. The following example shows how to use the formatter method for timeline cells.

INDEX.TS

```

import { Gantt } from '@syncfusion/ej2-gantt';
import { GanttData } from 'datasource.ts';
let gantt: Gantt = new Gantt({
  dataSource: GanttData,
  height: '450px',
  taskFields: {
    id: 'TaskID',
    name: 'TaskName',
    startDate: 'StartDate',
    duration: 'Duration',
    progress: 'Progress',
    child: 'subtasks'
  },
  timelineSettings: {
    timelineUnitSize: 50,
    topTier: {
      unit: "Month",
      format: "MMM dd, y"
    },
    bottomTier: {
      unit: "Day",
      formatter: date => {
        let presentDate = new Date(
          date.getFullYear(),
          date.getMonth(),
          date.getDate()
        );
        var start = new Date(presentDate.getFullYear(), 0, 0);
        var diff = Number(presentDate) - Number(start);
        var oneDay = 1000 * 60 * 60 * 24;
        var day = Math.floor(diff / oneDay);
        return day;
      }
    }
  },
  projectStartDate: new Date("01/01/2019"),
  projectEndDate: new Date("01/01/2020")
});

```

```
});
gantt.appendTo('#Gantt');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Gantt</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Gantt Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet" type="text/css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Gantt"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Grid

Getting started in EJ2 JavaScript Grid control

This section explains you the steps required to create a simple Essential JS 2 Grid and demonstrate the basic usage of the Grid control in a JavaScript application.

Dependencies

Following is the list of dependencies to use the grid with all features.

```
`javascript
|-- @syncfusion/ej2-grids
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-buttons
```

```
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-splitbuttons
|-- @syncfusion/ej2-calendars
|-- @syncfusion/ej2-excel-export
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-notifications
\
```

Setup for local environment

Refer the following steps for setup your local environment.

Step 1: Create a root folder **myapp** for your application.

Step 2: Create **myapp/resources** folder to store local scripts and styles files.

Step 3: Create **myapp/index.js** and **myapp/index.html** files for initializing Essential JS 2 Grid control.

Adding Syncfusion resources

The Essential JS 2 Grid control can be initialized by using either of the following ways.

- Using local script and style.
- Using CDN link for script and style.

Using local script and style

You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

After installing the Essential JS 2 product build, you can copy the grid and its dependencies scripts and style file into the **resources/scripts** and **resources/styles** folder.

Refer the below code to find location grid's script and style file.

Syntax:

Script: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js**

Styles: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css**

Example:

Script: C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-grids/dist/global/ej2-grids.min.js

Styles: C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-grids/styles/material.css

After copying the files, then you can refer the grid's scripts and styles into the `index.html` file.

The below html code example shows the minimal dependency of grid.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Grid</title>
<!-- Essential JS 2 Grid's dependent material theme -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/popups/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 material theme -->
<link href="resources/grids/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Grid's dependent scripts -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<!-- Essential JS 2 Grid's global script -->
<script src="resources/scripts/ej2-grids.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Using CDN link for script and style

Using CDN link, you can directly refer the grid control's script and style into the `index.html`.

Refer the grid's CDN links as below

Syntax:

Script: <http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js>

Styles: http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css

Example:

Script: <http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css>

The below html code example shows the minimal dependency of grid.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Grid</title>
<!-- Essential JS 2 Grid's dependent material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Grid's dependent script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-popups.min.js"
type="text/javascript"></script>
<!-- Essential JS 2 Grid's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js"
type="text/javascript"></script>
</head>
<body>
</body>
</html>
```

Adding Grid control

Now, you can start adding Grid control in the application. For getting started, add a **div** element for Grid control in **index.html**. Then refer the **index.js** file into the **index.html** file.

In this document context we are going to use **ej2.min.js** which includes all the Essential JS 2 components and its dependent scripts.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Grid</title>
<!-- Essential JS 2 Grid's dependent material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 all script -->
<script src="http://cdn.syncfusion.com/ej2/dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element for grid -->
<div id="Grid"></div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Place the following grid code in the **index.js**.

```
`javascript
var grid = new ej.grids.Grid();
grid.appendTo('#Grid');
```

Defining Row Data

Data for the Grid control is bind by using [dataSource](#) property. It accepts either array of JavaScript object or [DataManager](#) instance.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Grid</title>
<!-- Essential JS 2 Grid's dependent material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 all script -->
<script src="http://cdn.syncfusion.com/ej2/dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element for grid -->
<div id="Grid"></div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>
`
```

Place the following grid code in the **index.js**.

```
`javascript
var data = [{ OrderID: 10248, CustomerID: 'VINET', Freight: 32.38, OrderDate: new Date(8364186e5) },
{ OrderID: 10249, CustomerID: 'TOMSP', Freight: 11.61, OrderDate: new Date(836505e6) },
{ OrderID: 10250, CustomerID: 'HANAR', Freight: 65.83, OrderDate: new Date(8367642e5) },
{ OrderID: 10251, CustomerID: 'VICTE', Freight: 41.34, OrderDate: new Date(8368506e5) },
{ OrderID: 10252, CustomerID: 'SUPRD', Freight: 51.3, OrderDate: new Date(8367642e5) },
{ OrderID: 10253, CustomerID: 'HANAR', Freight: 58.17, OrderDate: new Date(836937e6) },
{ OrderID: 10254, CustomerID: 'CHOPS', Freight: 22.98, OrderDate: new Date(8370234e5) },
```

```
{ OrderID: 10255, CustomerID: 'RICSU', Freight: 148.33, OrderDate: new Date(8371098e5) },
{ OrderID: 10256, CustomerID: 'WELLI', Freight: 13.97, OrderDate: new Date(837369e6) }];
var grid = new ej.grids.Grid({dataSource: data});
grid.appendTo('#Grid');
`
```

Defining Columns

The Grid has an option to define columns as array. In these columns, we have properties to customize columns. Let's check the properties used here:

- We have added [field](#) to map with a property name an array of JavaScript objects.
- We have added [headerText](#) to change the title of columns.
- We have used [textAlign](#) to change the alignment of columns. By default, columns will be left aligned. To change columns to right align, we need to define [textAlign](#) as **Right**.
- Also, we have used another useful property, [format](#). Using this, we can format number and date values to standard or custom formats. Here, we have defined it for the conversion of numeric values to currency.

```
`javascript
var grid = new ej.grids.Grid({
dataSource: data,
columns: [
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 120, type: 'number' },
{ field: 'CustomerID', width: 140, headerText: 'Customer ID', type: 'string' },
{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C' },
{ field: 'OrderDate', headerText: 'Order Date', width: 140, format: 'yMd' }
]
});
grid.appendTo('#Grid');
`
```

Module injection

To create grids with additional features, inject the required modules. The following modules are used to extend grid's basic functionality.

- [Page](#) - Inject this module to use paging feature.
- [Sort](#) - Inject this module to use sorting feature.
- [Filter](#) - Inject this module to use filtering feature.
- [Group](#) - Inject this module to use grouping feature.
- **ExcelExport** - Inject this module to use Excel export feature.
- **PdfExport** - Inject this module to use PDF export feature.

These modules should be injected into the grid using the **`ej.grids.Grid.Inject`** method.

Additional feature modules are available [here](#).

Enable paging

The paging feature enables users to view the grid record in a paged view. It can be enabled by setting the [allowPaging](#) property to true. Inject the [Page](#) module as follows. If the [Page](#) module is not injected, the pager will not be rendered in the grid. Pager can be customized using the [pageSettings](#) property.

Create `my-app/es5-datasource.js` file to bind JSON data.

INDEX.JS

```
ej.grids.Grid.Inject(ej.grids.Page);
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ],
  allowPaging: true,
  pageSettings: { pageSize: 7 }
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ES5-DATASOURCE.JS

```

var data = [
    {
        OrderID: 10248, CustomerID: 'VINET', Role: 'Admin', EmployeeID: 5,
        OrderDate: new Date(8364186e5),
        ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims',
        ShipAddress: '59 rue de l Abbaye',
    }
]

```

```

        ShipRegion: 'CJ', Mask: '1111', ShipPostalCode: '51100', ShipCountry:
        'France', Freight: 32.38, Verified: !0
    },
    {
        OrderID: 10249, CustomerID: 'TOMSP', Role: 'Employee', EmployeeID:
        6, OrderDate: new Date(836505e6),
        ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress:
        'Luisenstr. 48',
        ShipRegion: 'CJ', Mask: '2222', ShipPostalCode: '44087',
        ShipCountry: 'Germany', Freight: 11.61, Verified: !1
    },
    {
        OrderID: 10250, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 4,
        OrderDate: new Date(8367642e5),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', Mask: '3333', ShipPostalCode: '05454-876',
        ShipCountry: 'Brazil', Freight: 65.83, Verified: !0
    },
    {
        OrderID: 10251, CustomerID: 'VICTE', Role: 'Manager', EmployeeID: 3,
        OrderDate: new Date(8367642e5),
        ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2,
        rue du Commerce',
        ShipRegion: 'CJ', Mask: '4444', ShipPostalCode: '69004',
        ShipCountry: 'France', Freight: 41.34, Verified: !0
    },
    {
        OrderID: 10252, CustomerID: 'SUPRD', Role: 'Manager', EmployeeID: 2,
        OrderDate: new Date(8368506e5),
        ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress:
        'Boulevard Tirou, 255',
        ShipRegion: 'CJ', Mask: '5555', ShipPostalCode: 'B-6000',
        ShipCountry: 'Belgium', Freight: 51.3, Verified: !0
    },
    {
        OrderID: 10253, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 7,
        OrderDate: new Date(836937e6),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', Mask: '6666', ShipPostalCode: '05454-876',
        ShipCountry: 'Brazil', Freight: 58.17, Verified: !0
    },
    {
        OrderID: 10254, CustomerID: 'CHOPS', Role: 'Employee', EmployeeID:
        5, OrderDate: new Date(8370234e5),
        ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress:
        'Hauptstr. 31',
        ShipRegion: 'CJ', Mask: '7777', ShipPostalCode: '3012', ShipCountry:
        'Switzerland', Freight: 22.98, Verified: !1
    },
    {
        OrderID: 10255, CustomerID: 'RICSU', Role: 'Admin', EmployeeID: 9,
        OrderDate: new Date(8371098e5),
        ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress:
        'Starenweg 5',

```

```

        ShipRegion: 'CJ', Mask: '8888', ShipPostalCode: '1204', ShipCountry:
        'Switzerland', Freight: 148.33, Verified: !0
    },
    {
        OrderID: 10256, CustomerID: 'WELLI', Role: 'Employee', EmployeeID:
        3, OrderDate: new Date(837369e6),
        ShipName: 'Wellington Importadora', ShipCity: 'Resende',
        ShipAddress: 'Rua do Mercado, 12',
        ShipRegion: 'SP', Mask: '9999', ShipPostalCode: '08737-363',
        ShipCountry: 'Brazil', Freight: 13.97, Verified: !1
    },
    {
        OrderID: 10257, CustomerID: 'HILAA', Role: 'Admin', EmployeeID: 4,
        OrderDate: new Date(8374554e5),
        ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal',
        ShipAddress: 'Carrera 22 con Ave. Carlos Soublette #8-35',
        ShipRegion: 'Táchira', Mask: '1234', ShipPostalCode: '5022',
        ShipCountry: 'Venezuela', Freight: 81.91, Verified: !0
    },
    {
        OrderID: 10258, CustomerID: 'ERNSH', Role: 'Manager', EmployeeID: 1,
        OrderDate: new Date(8375418e5),
        ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse
        6',
        ShipRegion: 'CJ', Mask: '2345', ShipPostalCode: '8010',
        ShipCountry: 'Austria', Freight: 140.51, Verified: !0
    },
    {
        OrderID: 10259, CustomerID: 'CENTC', Role: 'Admin', EmployeeID: 4,
        OrderDate: new Date(8376282e5),
        ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.',
        ShipAddress: 'Sierras de Granada 9993',
        ShipRegion: 'CJ', Mask: '3456', ShipPostalCode: '05022',
        ShipCountry: 'Mexico', Freight: 3.25, Verified: !1
    },
    {
        OrderID: 10260, CustomerID: 'OTTIK', Role: 'Admin', EmployeeID: 4,
        OrderDate: new Date(8377146e5),
        ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress:
        'Mehrheimerstr. 369',
        ShipRegion: 'CJ', Mask: '4567', ShipPostalCode: '50739',
        ShipCountry: 'Germany', Freight: 55.09, Verified: !0
    },
    {
        OrderID: 10261, CustomerID: 'QUEDE', Role: 'Manager', EmployeeID: 4,
        OrderDate: new Date(8377146e5),
        ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua da Panificadora, 12',
        ShipRegion: 'RJ', Mask: '5678', ShipPostalCode: '02389-673',
        ShipCountry: 'Brazil', Freight: 3.05, Verified: !1
    },
    {
        OrderID: 10262, CustomerID: 'RATTC', Role: 'Employee', EmployeeID:
        8, OrderDate: new Date(8379738e5),
        ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque',
        ShipAddress: '2817 Milton Dr.',

```



```

        ShipRegion: 'NM', Mask: '6789', ShipPostalCode: '87110',
        ShipCountry: 'USA', Freight: 48.29, Verified: !0
    }]];

```

Enable sorting

The sorting feature enables you to order the records. It can be enabled by setting the [allowSorting](#) property as true. Inject the [Sort](#) module as follows. If [Sort](#) module is not injected, you cannot sort when a header is clicked. Sorting feature can be customized using the [sortSettings](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Sort);
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ],
    allowSorting: true,
    allowPaging: true,
    pageSettings: { pageSize: 7 }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ES5-DATASOURCE.JS

```

var data = [
    {
        OrderID: 10248, CustomerID: 'VINET', Role: 'Admin', EmployeeID: 5,
        OrderDate: new Date(8364186e5),
        ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims',
        ShipAddress: '59 rue de l Abbaye',
    }
]

```

```

        ShipRegion: 'CJ', Mask: '1111', ShipPostalCode: '51100', ShipCountry:
        'France', Freight: 32.38, Verified: !0
    },
    {
        OrderID: 10249, CustomerID: 'TOMSP', Role: 'Employee', EmployeeID:
        6, OrderDate: new Date(836505e6),
        ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress:
        'Luisenstr. 48',
        ShipRegion: 'CJ', Mask: '2222', ShipPostalCode: '44087',
        ShipCountry: 'Germany', Freight: 11.61, Verified: !1
    },
    {
        OrderID: 10250, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 4,
        OrderDate: new Date(8367642e5),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', Mask: '3333', ShipPostalCode: '05454-876',
        ShipCountry: 'Brazil', Freight: 65.83, Verified: !0
    },
    {
        OrderID: 10251, CustomerID: 'VICTE', Role: 'Manager', EmployeeID: 3,
        OrderDate: new Date(8367642e5),
        ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2,
        rue du Commerce',
        ShipRegion: 'CJ', Mask: '4444', ShipPostalCode: '69004',
        ShipCountry: 'France', Freight: 41.34, Verified: !0
    },
    {
        OrderID: 10252, CustomerID: 'SUPRD', Role: 'Manager', EmployeeID: 2,
        OrderDate: new Date(8368506e5),
        ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress:
        'Boulevard Tirou, 255',
        ShipRegion: 'CJ', Mask: '5555', ShipPostalCode: 'B-6000',
        ShipCountry: 'Belgium', Freight: 51.3, Verified: !0
    },
    {
        OrderID: 10253, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 7,
        OrderDate: new Date(836937e6),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', Mask: '6666', ShipPostalCode: '05454-876',
        ShipCountry: 'Brazil', Freight: 58.17, Verified: !0
    },
    {
        OrderID: 10254, CustomerID: 'CHOPS', Role: 'Employee', EmployeeID:
        5, OrderDate: new Date(8370234e5),
        ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress:
        'Hauptstr. 31',
        ShipRegion: 'CJ', Mask: '7777', ShipPostalCode: '3012', ShipCountry:
        'Switzerland', Freight: 22.98, Verified: !1
    },
    {
        OrderID: 10255, CustomerID: 'RICSU', Role: 'Admin', EmployeeID: 9,
        OrderDate: new Date(8371098e5),
        ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress:
        'Starenweg 5',

```

```

        ShipRegion: 'CJ', Mask: '8888', ShipPostalCode: '1204', ShipCountry:
'Switzerland', Freight: 148.33, Verified: !0
    },
    {
        OrderID: 10256, CustomerID: 'WELLI', Role: 'Employee', EmployeeID:
3, OrderDate: new Date(837369e6),
        ShipName: 'Wellington Importadora', ShipCity: 'Resende',
ShipAddress: 'Rua do Mercado, 12',
        ShipRegion: 'SP', Mask: '9999', ShipPostalCode: '08737-363',
ShipCountry: 'Brazil', Freight: 13.97, Verified: !1
    },
    {
        OrderID: 10257, CustomerID: 'HILAA', Role: 'Admin', EmployeeID: 4,
OrderDate: new Date(8374554e5),
        ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal',
ShipAddress: 'Carrera 22 con Ave. Carlos Soublette #8-35',
        ShipRegion: 'Táchira', Mask: '1234', ShipPostalCode: '5022',
ShipCountry: 'Venezuela', Freight: 81.91, Verified: !0
    },
    {
        OrderID: 10258, CustomerID: 'ERNSH', Role: 'Manager', EmployeeID: 1,
OrderDate: new Date(8375418e5),
        ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse
6',
        ShipRegion: 'CJ', Mask: '2345', ShipPostalCode: '8010',
ShipCountry: 'Austria', Freight: 140.51, Verified: !0
    },
    {
        OrderID: 10259, CustomerID: 'CENTC', Role: 'Admin', EmployeeID: 4,
OrderDate: new Date(8376282e5),
        ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.',
ShipAddress: 'Sierras de Granada 9993',
        ShipRegion: 'CJ', Mask: '3456', ShipPostalCode: '05022',
ShipCountry: 'Mexico', Freight: 3.25, Verified: !1
    },
    {
        OrderID: 10260, CustomerID: 'OTTIK', Role: 'Admin', EmployeeID: 4,
OrderDate: new Date(8377146e5),
        ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress:
'Mehrheimerstr. 369',
        ShipRegion: 'CJ', Mask: '4567', ShipPostalCode: '50739',
ShipCountry: 'Germany', Freight: 55.09, Verified: !0
    },
    {
        OrderID: 10261, CustomerID: 'QUEDE', Role: 'Manager', EmployeeID: 4,
OrderDate: new Date(8377146e5),
        ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua da Panificadora, 12',
        ShipRegion: 'RJ', Mask: '5678', ShipPostalCode: '02389-673',
ShipCountry: 'Brazil', Freight: 3.05, Verified: !1
    },
    {
        OrderID: 10262, CustomerID: 'RATTC', Role: 'Employee', EmployeeID:
8, OrderDate: new Date(8379738e5),
        ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque',
ShipAddress: '2817 Milton Dr.',

```

```

        ShipRegion: 'NM', Mask: '6789', ShipPostalCode: '87110',
        ShipCountry: 'USA', Freight: 48.29, Verified: !0
    }];

```

Enable filtering

The filtering feature enables you to view reduced amount of records based on filter criteria. It can be enabled by setting the [allowFiltering](#) property as true. The [Filter](#) module has to be injected as follows. If [Filter](#) module is not injected, filter bar will not be rendered in the grid. Filtering feature can be customized using the [filterSettings](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Sort, ej.grids.Filter);
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ],
    allowFiltering: true,
    allowPaging: true,
    pageSettings: { pageSize: 6 },
    allowSorting: true
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ES5-DATASOURCE.JS

```

var data = [
    {
        OrderID: 10248, CustomerID: 'VINET', Role: 'Admin', EmployeeID: 5,
        OrderDate: new Date(8364186e5),
    }
]

```

```

        ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims',
        ShipAddress: '59 rue de l Abbaye',
        ShipRegion: 'CJ', Mask: '1111', ShipPostalCode: '51100', ShipCountry:
        'France', Freight: 32.38, Verified: !0
    },
    {
        OrderID: 10249, CustomerID: 'TOMSP', Role: 'Employee', EmployeeID:
        6, OrderDate: new Date(836505e6),
        ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress:
        'Luisenstr. 48',
        ShipRegion: 'CJ', Mask: '2222', ShipPostalCode: '44087',
        ShipCountry: 'Germany', Freight: 11.61, Verified: !1
    },
    {
        OrderID: 10250, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 4,
        OrderDate: new Date(8367642e5),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', Mask: '3333', ShipPostalCode: '05454-876',
        ShipCountry: 'Brazil', Freight: 65.83, Verified: !0
    },
    {
        OrderID: 10251, CustomerID: 'VICTE', Role: 'Manager', EmployeeID: 3,
        OrderDate: new Date(8367642e5),
        ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2,
        rue du Commerce',
        ShipRegion: 'CJ', Mask: '4444', ShipPostalCode: '69004',
        ShipCountry: 'France', Freight: 41.34, Verified: !0
    },
    {
        OrderID: 10252, CustomerID: 'SUPRD', Role: 'Manager', EmployeeID: 2,
        OrderDate: new Date(8368506e5),
        ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress:
        'Boulevard Tirou, 255',
        ShipRegion: 'CJ', Mask: '5555', ShipPostalCode: 'B-6000',
        ShipCountry: 'Belgium', Freight: 51.3, Verified: !0
    },
    {
        OrderID: 10253, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 7,
        OrderDate: new Date(836937e6),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', Mask: '6666', ShipPostalCode: '05454-876',
        ShipCountry: 'Brazil', Freight: 58.17, Verified: !0
    },
    {
        OrderID: 10254, CustomerID: 'CHOPS', Role: 'Employee', EmployeeID:
        5, OrderDate: new Date(8370234e5),
        ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress:
        'Hauptstr. 31',
        ShipRegion: 'CJ', Mask: '7777', ShipPostalCode: '3012', ShipCountry:
        'Switzerland', Freight: 22.98, Verified: !1
    },
    {
        OrderID: 10255, CustomerID: 'RICSU', Role: 'Admin', EmployeeID: 9,
        OrderDate: new Date(8371098e5),

```

```

        ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress:
'Starenweg 5',
        ShipRegion: 'CJ', Mask: '8888', ShipPostalCode: '1204', ShipCountry:
'Switzerland', Freight: 148.33, Verified: !0
    },
    {
        OrderID: 10256, CustomerID: 'WELLI', Role: 'Employee', EmployeeID:
3, OrderDate: new Date(837369e6),
        ShipName: 'Wellington Importadora', ShipCity: 'Resende',
ShipAddress: 'Rua do Mercado, 12',
        ShipRegion: 'SP', Mask: '9999', ShipPostalCode: '08737-363',
ShipCountry: 'Brazil', Freight: 13.97, Verified: !1
    },
    {
        OrderID: 10257, CustomerID: 'HILAA', Role: 'Admin', EmployeeID: 4,
OrderDate: new Date(8374554e5),
        ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal',
ShipAddress: 'Carrera 22 con Ave. Carlos Soublette #8-35',
        ShipRegion: 'Táchira', Mask: '1234', ShipPostalCode: '5022',
ShipCountry: 'Venezuela', Freight: 81.91, Verified: !0
    },
    {
        OrderID: 10258, CustomerID: 'ERNSH', Role: 'Manager', EmployeeID: 1,
OrderDate: new Date(8375418e5),
        ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse
6',
        ShipRegion: 'CJ', Mask: '2345', ShipPostalCode: '8010',
ShipCountry: 'Austria', Freight: 140.51, Verified: !0
    },
    {
        OrderID: 10259, CustomerID: 'CENTC', Role: 'Admin', EmployeeID: 4,
OrderDate: new Date(8376282e5),
        ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.',
ShipAddress: 'Sierras de Granada 9993',
        ShipRegion: 'CJ', Mask: '3456', ShipPostalCode: '05022',
ShipCountry: 'Mexico', Freight: 3.25, Verified: !1
    },
    {
        OrderID: 10260, CustomerID: 'OTTIK', Role: 'Admin', EmployeeID: 4,
OrderDate: new Date(8377146e5),
        ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress:
'Mehrheimerstr. 369',
        ShipRegion: 'CJ', Mask: '4567', ShipPostalCode: '50739',
ShipCountry: 'Germany', Freight: 55.09, Verified: !0
    },
    {
        OrderID: 10261, CustomerID: 'QUEDE', Role: 'Manager', EmployeeID: 4,
OrderDate: new Date(8377146e5),
        ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua da Panificadora, 12',
        ShipRegion: 'RJ', Mask: '5678', ShipPostalCode: '02389-673',
ShipCountry: 'Brazil', Freight: 3.05, Verified: !1
    },
    {
        OrderID: 10262, CustomerID: 'RATTC', Role: 'Employee', EmployeeID:
8, OrderDate: new Date(8379738e5),

```



```

        ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque',
        ShipAddress: '2817 Milton Dr.',
        ShipRegion: 'NM', Mask: '6789', ShipPostalCode: '87110',
        ShipCountry: 'USA', Freight: 48.29, Verified: !0
    }];

```

Enable grouping

The grouping feature enables users to view the grid record in a grouped view. It can be enabled by setting the [allowGrouping](#) property to true. The [Group](#) module has to be injected as follows. If [Group](#) module is not injected, the group drop area will not be rendered in the grid. Grouping feature can be customized using the [groupSettings](#) property.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Sort, ej.grids.Filter,
ej.grids.Group);
var grid = new ej.grids.Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ],
    height: 180,
    pageSettings: { pageSize: 7 },
    allowGrouping: true,
    allowPaging: true,
    allowSorting: true,
    allowFiltering: true
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ES5-DATASOURCE.JS

```

var data = [
    {
        OrderID: 10248, CustomerID: 'VINET', Role: 'Admin', EmployeeID: 5,
        OrderDate: new Date(8364186e5),
        ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims',
        ShipAddress: '59 rue de l Abbaye',
        ShipRegion: 'CJ', Mask: '1111', ShipPostalCode: '51100', ShipCountry:
        'France', Freight: 32.38, Verified: !0
    },
    {
        OrderID: 10249, CustomerID: 'TOMSP', Role: 'Employee', EmployeeID:
        6, OrderDate: new Date(836505e6),
        ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress:
        'Luisenstr. 48',
        ShipRegion: 'CJ', Mask: '2222', ShipPostalCode: '44087',
        ShipCountry: 'Germany', Freight: 11.61, Verified: !1
    },
    {
        OrderID: 10250, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 4,
        OrderDate: new Date(8367642e5),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', Mask: '3333', ShipPostalCode: '05454-876',
        ShipCountry: 'Brazil', Freight: 65.83, Verified: !0
    },
    {
        OrderID: 10251, CustomerID: 'VICTE', Role: 'Manager', EmployeeID: 3,
        OrderDate: new Date(8367642e5),
        ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2,
        rue du Commerce',
        ShipRegion: 'CJ', Mask: '4444', ShipPostalCode: '69004',
        ShipCountry: 'France', Freight: 41.34, Verified: !0
    },
    {
        OrderID: 10252, CustomerID: 'SUPRD', Role: 'Manager', EmployeeID: 2,
        OrderDate: new Date(8368506e5),
        ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress:
        'Boulevard Tirou, 255',
        ShipRegion: 'CJ', Mask: '5555', ShipPostalCode: 'B-6000',
        ShipCountry: 'Belgium', Freight: 51.3, Verified: !0
    },
    {
        OrderID: 10253, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 7,
        OrderDate: new Date(836937e6),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', Mask: '6666', ShipPostalCode: '05454-876',
        ShipCountry: 'Brazil', Freight: 58.17, Verified: !0
    },
    {
        OrderID: 10254, CustomerID: 'CHOPS', Role: 'Employee', EmployeeID:
        5, OrderDate: new Date(8370234e5),
        ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress:
        'Hauptstr. 31',
        ShipRegion: 'CJ', Mask: '7777', ShipPostalCode: '3012', ShipCountry:
        'Switzerland', Freight: 22.98, Verified: !1
    },
]

```

```

{
    OrderID: 10255, CustomerID: 'RICSU', Role: 'Admin', EmployeeID: 9,
    OrderDate: new Date(8371098e5),
    ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress:
    'Starenweg 5',
    ShipRegion: 'CJ', Mask: '8888', ShipPostalCode: '1204', ShipCountry:
    'Switzerland', Freight: 148.33, Verified: !0
},
{
    OrderID: 10256, CustomerID: 'WELLI', Role: 'Employee', EmployeeID:
    3, OrderDate: new Date(837369e6),
    ShipName: 'Wellington Importadora', ShipCity: 'Resende',
    ShipAddress: 'Rua do Mercado, 12',
    ShipRegion: 'SP', Mask: '9999', ShipPostalCode: '08737-363',
    ShipCountry: 'Brazil', Freight: 13.97, Verified: !1
},
{
    OrderID: 10257, CustomerID: 'HILAA', Role: 'Admin', EmployeeID: 4,
    OrderDate: new Date(8374554e5),
    ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal',
    ShipAddress: 'Carrera 22 con Ave. Carlos Soublette #8-35',
    ShipRegion: 'Táchira', Mask: '1234', ShipPostalCode: '5022',
    ShipCountry: 'Venezuela', Freight: 81.91, Verified: !0
},
{
    OrderID: 10258, CustomerID: 'ERNSH', Role: 'Manager', EmployeeID: 1,
    OrderDate: new Date(8375418e5),
    ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse
    6',
    ShipRegion: 'CJ', Mask: '2345', ShipPostalCode: '8010',
    ShipCountry: 'Austria', Freight: 140.51, Verified: !0
},
{
    OrderID: 10259, CustomerID: 'CENTC', Role: 'Admin', EmployeeID: 4,
    OrderDate: new Date(8376282e5),
    ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.',
    ShipAddress: 'Sierras de Granada 9993',
    ShipRegion: 'CJ', Mask: '3456', ShipPostalCode: '05022',
    ShipCountry: 'Mexico', Freight: 3.25, Verified: !1
},
{
    OrderID: 10260, CustomerID: 'OTTIK', Role: 'Admin', EmployeeID: 4,
    OrderDate: new Date(8377146e5),
    ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress:
    'Mehrheimerstr. 369',
    ShipRegion: 'CJ', Mask: '4567', ShipPostalCode: '50739',
    ShipCountry: 'Germany', Freight: 55.09, Verified: !0
},
{
    OrderID: 10261, CustomerID: 'QUEDE', Role: 'Manager', EmployeeID: 4,
    OrderDate: new Date(8377146e5),
    ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress:
    'Rua da Panificadora, 12',
    ShipRegion: 'RJ', Mask: '5678', ShipPostalCode: '02389-673',
    ShipCountry: 'Brazil', Freight: 3.05, Verified: !1
},
{

```

```

      OrderID: 10262, CustomerID: 'RATTC', Role: 'Employee', EmployeeID:
8, OrderDate: new Date(8379738e5),
      ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque',
ShipAddress: '2817 Milton Dr.',
      ShipRegion: 'NM', Mask: '6789', ShipPostalCode: '87110',
ShipCountry: 'USA', Freight: 48.29, Verified: !0
    }
  ];

```

Run the application

Now, run the **index.html** in web browser, it will render the Essential JS 2 Grid control.

Output will be displayed as follows.

INDEX.JS

```

ej.grids.Grid.Inject(ej.grids.Page, ej.grids.Sort, ej.grids.Filter,
ej.grids.Group);
var grid = new ej.grids.Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ],
  height: 175,
  pageSettings: { pageSize: 7 },
  allowGrouping: true,
  allowPaging: true,
  allowSorting: true,
  allowFiltering: true
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ES5-DATASOURCE.JS

```

var data = [
    {
        OrderID: 10248, CustomerID: 'VINET', Role: 'Admin', EmployeeID: 5,
        OrderDate: new Date(8364186e5),
        ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims',
        ShipAddress: '59 rue de l Abbaye',
        ShipRegion: 'CJ', Mask: '1111', ShipPostalCode: '51100', ShipCountry:
        'France', Freight: 32.38, Verified: !0
    },
    {
        OrderID: 10249, CustomerID: 'TOMSP', Role: 'Employee', EmployeeID:
        6, OrderDate: new Date(836505e6),
        ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress:
        'Luisenstr. 48',
        ShipRegion: 'CJ', Mask: '2222', ShipPostalCode: '44087',
        ShipCountry: 'Germany', Freight: 11.61, Verified: !1
    },
    {
        OrderID: 10250, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 4,
        OrderDate: new Date(8367642e5),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', Mask: '3333', ShipPostalCode: '05454-876',
        ShipCountry: 'Brazil', Freight: 65.83, Verified: !0
    },
    {
        OrderID: 10251, CustomerID: 'VICTE', Role: 'Manager', EmployeeID: 3,
        OrderDate: new Date(8367642e5),
        ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2,
        rue du Commerce',
        ShipRegion: 'CJ', Mask: '4444', ShipPostalCode: '69004',
        ShipCountry: 'France', Freight: 41.34, Verified: !0
    },
    {
        OrderID: 10252, CustomerID: 'SUPRD', Role: 'Manager', EmployeeID: 2,
        OrderDate: new Date(8368506e5),
        ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress:
        'Boulevard Tirou, 255',
        ShipRegion: 'CJ', Mask: '5555', ShipPostalCode: 'B-6000',
        ShipCountry: 'Belgium', Freight: 51.3, Verified: !0
    },
    {
        OrderID: 10253, CustomerID: 'HANAR', Role: 'Admin', EmployeeID: 7,
        OrderDate: new Date(836937e6),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', Mask: '6666', ShipPostalCode: '05454-876',
        ShipCountry: 'Brazil', Freight: 58.17, Verified: !0
    },
    {
        OrderID: 10254, CustomerID: 'CHOPS', Role: 'Employee', EmployeeID:
        5, OrderDate: new Date(8370234e5),
        ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress:
        'Hauptstr. 31',
        ShipRegion: 'CJ', Mask: '7777', ShipPostalCode: '3012', ShipCountry:
        'Switzerland', Freight: 22.98, Verified: !1
    },
]

```

```

{
    OrderID: 10255, CustomerID: 'RICSU', Role: 'Admin', EmployeeID: 9,
    OrderDate: new Date(8371098e5),
    ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress:
    'Starenweg 5',
    ShipRegion: 'CJ', Mask: '8888', ShipPostalCode: '1204', ShipCountry:
    'Switzerland', Freight: 148.33, Verified: !0
},
{
    OrderID: 10256, CustomerID: 'WELLI', Role: 'Employee', EmployeeID:
    3, OrderDate: new Date(837369e6),
    ShipName: 'Wellington Importadora', ShipCity: 'Resende',
    ShipAddress: 'Rua do Mercado, 12',
    ShipRegion: 'SP', Mask: '9999', ShipPostalCode: '08737-363',
    ShipCountry: 'Brazil', Freight: 13.97, Verified: !1
},
{
    OrderID: 10257, CustomerID: 'HILAA', Role: 'Admin', EmployeeID: 4,
    OrderDate: new Date(8374554e5),
    ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal',
    ShipAddress: 'Carrera 22 con Ave. Carlos Soublette #8-35',
    ShipRegion: 'Táchira', Mask: '1234', ShipPostalCode: '5022',
    ShipCountry: 'Venezuela', Freight: 81.91, Verified: !0
},
{
    OrderID: 10258, CustomerID: 'ERNSH', Role: 'Manager', EmployeeID: 1,
    OrderDate: new Date(8375418e5),
    ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse
    6',
    ShipRegion: 'CJ', Mask: '2345', ShipPostalCode: '8010',
    ShipCountry: 'Austria', Freight: 140.51, Verified: !0
},
{
    OrderID: 10259, CustomerID: 'CENTC', Role: 'Admin', EmployeeID: 4,
    OrderDate: new Date(8376282e5),
    ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.',
    ShipAddress: 'Sierras de Granada 9993',
    ShipRegion: 'CJ', Mask: '3456', ShipPostalCode: '05022',
    ShipCountry: 'Mexico', Freight: 3.25, Verified: !1
},
{
    OrderID: 10260, CustomerID: 'OTTIK', Role: 'Admin', EmployeeID: 4,
    OrderDate: new Date(8377146e5),
    ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress:
    'Mehrheimerstr. 369',
    ShipRegion: 'CJ', Mask: '4567', ShipPostalCode: '50739',
    ShipCountry: 'Germany', Freight: 55.09, Verified: !0
},
{
    OrderID: 10261, CustomerID: 'QUEDE', Role: 'Manager', EmployeeID: 4,
    OrderDate: new Date(8377146e5),
    ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress:
    'Rua da Panificadora, 12',
    ShipRegion: 'RJ', Mask: '5678', ShipPostalCode: '02389-673',
    ShipCountry: 'Brazil', Freight: 3.05, Verified: !1
},
{

```



```

      OrderID: 10262, CustomerID: 'RATTC', Role: 'Employee', EmployeeID:
      8, OrderDate: new Date(8379738e5),
      ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque',
      ShipAddress: '2817 Milton Dr.',
      ShipRegion: 'NM', Mask: '6789', ShipPostalCode: '87110',
      ShipCountry: 'USA', Freight: 48.29, Verified: !0
    }
  ];

```

Deploy the Application

The Essential JS 2 Grid control features are segregated into individual feature-wise modules. The [Essential Studio JavaScript \(Essential JS 2\)](#) build and **CDN** scripts contains code for all features used in grid and hence we suggest to not to use them in production. We strongly recommend you to generate script files to use in production using our **Custom Resource Generator**([CRG](#)) for Essential JS 2. CRG will allow you to generate the bundled script for the currently enabled features in grid.

See Also

- [How to display a table data after clicking Submit button in Javascript?](#)
- [How to display table in popup window using Javascript?](#)
- [How to open pdf document on button click inside a Grid](#)
- [How to disable the default keyboard actions in Grid](#)

Module in EJ2 JavaScript Grid control

The following feature modules should be injected to extend grid's functionality.

The available grid modules are:

Module	Description
----- -----	
Page	Inject this module to use paging feature.
Sort	Inject this module to use sorting feature.
Filter	Inject this module to use filtering feature.
Group	Inject this module to use grouping feature
Edit	Inject this module is use editing feature.
Aggregate	Inject this module to use aggregate feature.
ColumnChooser	Inject this module to use column chooser feature.
ColumnMenu	Inject this module to use column menu feature.
CommandColumn	Inject this module to use command column feature.
ContextMenu	Inject this module to use context menu feature.
DetailRow	Inject this module to use detail template feature.
ForeignKey	Inject this module to use foreign key feature.
Freeze	Inject this module to use frozen rows and columns feature.

- | **Resize** | Inject this module to use resize feature. |
- | [Reorder](#) | Inject this module to use reorder feature. |
- | **RowDD** | Inject this module to use row drag and drop feature. |
- | [Search](#) | Inject this module to use search feature and this is a default injected module. |
- | [Selection](#) | Inject this module to use selection feature and this is a default injected module. |
- | [Scroll](#) | Inject this module to use scrolling feature and this is a default injected module. |
- | [Print](#) | Inject this module to use to use print feature and this is a default injected module. |
- | [Toolbar](#) | Inject this module to use toolbar feature. |
- | **VirtualScroll** | Inject this module to use virtual scroll feature. |
- | [ExcelExport](#) | Inject this module to use excel export feature. |
- | [PdfExport](#) | Inject this module to use PDF export feature. |

These modules should be injected into the grid using the **Grid.Inject** method.

Data Binding

Data binding in EJ2 JavaScript Grid control

The Grid uses [DataManager](#), which supports both RESTful JSON data services binding and local JavaScript object array binding. The [dataSource](#) property can be assigned either with the instance of [DataManager](#) or JavaScript object array collection.

It supports two kinds of data binding method:

- Local data
- Remote data

Sending additional parameters to the server

To add a custom parameter to the data request, use the **addParams** method of **Query** class. Assign the **Query** object with additional parameters to the grid [query](#) property.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { DataManager, Query, ODataAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://services.syncfusion.com/js/production/api/Orders',
    adaptor: new ODataAdaptor
});
let grid: Grid = new Grid({
    dataSource: data,
    query: new Query().addParams('ej2grid', 'true'),
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
```

```

        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>

```

```

    }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The parameters added using the [query](#) property will be sent along with the data request for every grid action.

Handling HTTP error

During server interaction from the grid, some server-side exceptions may occur, and you can acquire those error messages or exception details

in client-side using the [actionFailure](#) event.

The argument passed to the [actionFailure](#) event contains the error details returned from the server.

The [actionFailure](#) event will be triggered not only for the server errors, but also when there is an exception while processing the grid actions.

Binding with ajax

You can use Grid [dataSource](#) property to bind the datasource to Grid from external ajax request. In the below code we have fetched the datasource from the server with the help of ajax request and provided that to [dataSource](#) property by using **onSuccess** event of the ajax.

```

`ts
import { Grid, Page } from '@syncfusion/ej2-grids';
import { Ajax } from '@syncfusion/ej2-base';

Grid.Inject(Page);

let grid: Grid = new Grid({
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', textAlign: 'Right', width: 120 },

```

```

{ field: 'EmployeeID', headerText: 'Employee ID', textAlign: 'Right', width: 120 },
{ field: 'ShipCountry', headerText: 'Ship Country', textAlign: 'Right', width: 120 }
]
});
grid.appendTo('#Grid');
let button = document.getElementById('btn');
button.addEventListener("click", function(e){
let ajax = new Ajax("https://ej2services.syncfusion.com/production/web-services/api/Orders", "GET");
ajax.send();
ajax.onSuccess = function (data: string) {
grid.dataSource = JSON.parse(data);
};
});
`

```

* If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server side crud actions.

See Also

- [How to bind SQL Server data in TypeScript DataGrid using SqlClient data provider](#)

Local data in EJ2 JavaScript Grid control

To bind local data to the grid, you can assign a JavaScript object array to the [dataSource](#) property. The local data source can also be provided as an instance of the [DataManager](#).

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let gridData: Object[] = data.slice(0, 7);
let grid: Grid = new Grid({
  dataSource: gridData,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, [DataManager](#) uses **JsonAdaptor** for local data-binding.

Refresh the data source

You can add/delete the data source records through an external button. To reflect the data source changes in the grid, invoke the [refresh](#) method.

To refresh the data source:

Step 1:

Add/delete the data source record by using the following code.

```

`ts
grid.dataSource.unshift(data); // add a new record.
grid.dataSource.splice(selectedRow, 1); // delete a record.
`

```

Step 2:

Refresh the grid after the data source change by using the [refresh](#) method.

```

`ts
grid.refresh(); // refresh the Grid.
`

```

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },

```

```

        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 280
});
grid.appendTo('#Grid');
let add: Button = new Button({}, '#add');
let dele: Button = new Button({}, '#delete');
document.getElementById('add').onclick = () => {
    let data: Object = { OrderID: 10247, CustomerID: "ASDFG", Freight: 40.4,
OrderDate: new Date(8367642e5) };
    (<Object[]>grid.dataSource).unshift(data); // add new record.
    grid.refresh(); // refresh the Grid.
};
document.getElementById('delete').onclick = () => {
    if (grid.getSelectedRowIndex().length) {
        let selectedRow: number = grid.getSelectedRowIndex()[0];
        (<Object[]>grid.dataSource).splice(selectedRow, 1); // delete the
selected record.
    } else {
        alert("No records selected for delete operation");
    }
    grid.refresh(); // refresh the Grid.
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button class="e-flat" id="add">Add</button>
        <button class="e-flat" id="delete">Delete</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Remote data in EJ2 JavaScript Grid control

To bind remote data to grid component, assign service data as an instance of [DataManager](#) to the [dataSource](#) property. To interact with remote data source, provide the endpoint [url](#).

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://services.syncfusion.com/js/production/api/Orders',
    adaptor: new ODataAdaptor
});
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, [DataManager](#) uses **ODataAdaptor** for remote data-binding.

OData adaptor - Binding OData service

[OData](#) is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the DataManager. Refer to the following code example for remote Data binding using OData service.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://services.syncfusion.com/js/production/api/Orders',
    adaptor: new ODataAdaptor,
    crossDomain: true
});
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

OData v4 adaptor - Binding OData v4 service

The ODataV4 is an improved version of OData protocols, and the [DataManager](#) can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the [odata documentation](#). To bind OData v4 service, use the **ODataV4Adaptor**.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/?$top=7',
    adaptor: new ODataV4Adaptor
});
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
        'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
        ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
        'Right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
        format: 'yMd' }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    richtexteditor/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Web API adaptor

You can use **WebApiAdaptor** to bind grid with Web API created using OData endpoint.

```

`ts
import { Grid } from '@syncfusion/ej2-grids';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
url: '/api/OrderAPI',
adaptor: new WebApiAdaptor
});
let grid: Grid = new Grid({
dataSource: data,
columns: [
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 120, type: 'number' },
{ field: 'CustomerID', width: 140, headerText: 'Customer ID', type: 'string' },
{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C' },
{ field: 'OrderDate', headerText: 'Order Date', width: 140, format: 'yMd' }
]
});
grid.appendTo('#Grid');
`

```

The response object should contain properties, **Items** and **Count**, whose values are a collection of entities and total count of the entities, respectively.

The sample response object should look like this:

```

`
{
Items: [{..}, {..}, {..}, ...],
Count: 830
}
`

```

Remote save adaptor

You may need to perform all Grid Actions in client-side except the CRUD operations, that should be interacted with server-side to persist data. It can be achieved in Grid by using **RemoteSaveAdaptor**.

Datasource must be set to **json** property and set **RemoteSaveAdaptor** to the **adaptor** property. CRUD operations can be mapped to server-side using **updateUrl**, **insertUrl**, **removeUrl**, **batchUrl**, **crudUrl** properties.

You can use the following code example to use **RemoteSaveAdaptor** in Grid.

```

`ts

```

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, RemoteSaveAdaptor } from '@syncfusion/ej2-data';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let dataSource: DataManager = new DataManager({
  json: data,
  adaptor: new RemoteSaveAdaptor,
  insertUrl: '/Home/Insert',
  updateUrl: '/Home/Update',
  removeUrl: '/Home/Delete'
});
let grid: Grid = new Grid({
  dataSource: dataSource,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', isPrimaryKey: true, textAlign: 'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140, format: 'yMd' }
  ]
});
grid.appendTo('#Grid');

```

The following code example describes the CRUD operations handled at server-side.

```

public ActionResult Update(OrdersDetails value)
{
  ...
  return Json(value);
}

public ActionResult Insert(OrdersDetails value)

```



```

{
...
return Json(value);
}

public ActionResult Delete(int key)
{
...
return Json(data);
}
,

```

Custom adaptor

You can create your own adaptor by extending the built-in adaptors. The following demonstrates custom adaptor approach and how to add a serial number for the records by overriding the built-in response processing using the **processResponse** method of the **ODataAdaptor**.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
class SerialNoAdaptor extends ODataAdaptor {
    processResponse(): Object {
        let i: number = 0;
        // calling base class processResponse function
        let original: {result: Object[], count: number} =
super.processResponse.apply(this, arguments);
        // adding serial number
        original.result.forEach((item: Object) => item['Sno'] = ++i);
        return { result: original.result, count: original.count };
    }
}
let data: DataManager = new DataManager({
    url: 'https://services.syncfusion.com/js/production/api/Orders',
    adaptor: new SerialNoAdaptor
});
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'Sno', width: 120, headerText: 'SNO', textAlign: 'Right' },
        { field: 'CustomerID', width: 140, headerText: 'Customer Name',
type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd' }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Offline mode

On remote data binding, all grid actions such as paging, sorting, editing, grouping, filtering, etc, will be processed on server-side. To avoid postback for every action, set the grid to load all data on initialization and make the actions process in client-side. To enable this behavior, use the **offline** property of [DataManager](#).

INDEX.TS

```

import { Grid, Page, Sort, Group } from '@syncfusion/ej2-grids';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Page, Sort, Group);
let data: DataManager = new DataManager({
  url: 'https://services.syncfusion.com/js/production/api/Orders',
  adaptor: new ODataAdaptor,
  offline: true
});
let grid: Grid = new Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ],
  allowPaging: true,
  pageSettings: { pageSize: 7 },
  allowSorting: true,
  allowGrouping: true
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">

```

```

    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Immutable mode in EJ2 JavaScript Grid control

The immutable mode optimizes the Grid re-rendering performance by using the object reference and [deep compare](#) concept. When performing the Grid actions, it will only re-render the modified or newly added rows and prevent the re-rendering of the unchanged rows.

To enable this feature, you have to set the [enableImmutableMode](#) property as **true**.

* This feature uses the primary key value for data comparison. So, you need to provide the [isPrimaryKey](#) column.

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
Grid.Inject(Page);
let immutableStart;
let normalStart;
let primaryKey = 0;
let immutableInit = true;
let init = true;
let immutableGrid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    pageSettings: { pageSize: 50 },
    enableImmutableMode: true,
    beforeDataBound: () => {
        immutableStart = new Date().getTime();
    },
    dataBound: () => {
        let val = immutableInit ? '' : new Date().getTime() -
immutableStart;
        document.getElementById('immutableDelete').innerHTML = 'Immutable
rendering Time: ' + "<b>" + val + "</b>" + '<b>ms</b>';
        immutableStart = 0; immutableInit = false;
    },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', isPrimaryKey: true,
textAlign: 'Right', width: 120 },
        { field: 'ProductName', headerText: 'Product Name', width: 160 },
        { field: 'ProductID', headerText: 'Product ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'CustomerName', headerText: 'Customer Name', width: 160 }
    ],
    height: 240
});
immutableGrid.appendTo('#immutable');

```

```

document.getElementById('delete').addEventListener('click', function(e) {
    (immutableGrid.dataSource as Object[]).splice(0, 5);
    normalGrid.setProperties({ dataSource: immutableGrid.dataSource });
    immutableGrid.setProperties({ dataSource: immutableGrid.dataSource });
});
document.getElementById('addtop').addEventListener('click', function(e) {
    let addedRecords = [
        { 'OrderID': ++primaryKey, 'ProductName': 'Chai', 'ProductID':
'Sasquatch Ale', 'CustomerID': 'QUEDE', 'CustomerName': 'Yoshi Tannamuri' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Georg Pippis',
'ProductID': 'Valkoinen suklaa', 'CustomerID': 'RATTC', 'CustomerName':
'Martín Sommer' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Yoshi Tannamuri',
'ProductID': 'Gula Malacca', 'CustomerID': 'COMMI', 'CustomerName': 'Ann
Devon' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Palle Ibsen',
'ProductID': 'Rogede sild', 'CustomerID': 'RATTC', 'CustomerName': 'Paula
Wilson' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Francisco Chang',
'ProductID': 'Mascarpone Fabioli', 'CustomerID': 'ROMEY', 'CustomerName':
'Jose Pavarotti' }
    ]
    let aData = (addedRecords as Object[]).concat(immutableGrid.dataSource);
    normalGrid.setProperties({ dataSource: aData });
    immutableGrid.setProperties({ dataSource: aData });
});
document.getElementById('addbottom').addEventListener('click', function(e) {
    let addedRecords = [
        { 'OrderID': ++primaryKey, 'ProductName': 'Chai', 'ProductID':
'Sasquatch Ale', 'CustomerID': 'QUEDE', 'CustomerName': 'Yoshi Tannamuri' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Georg Pippis',
'ProductID': 'Valkoinen suklaa', 'CustomerID': 'RATTC', 'CustomerName':
'Martín Sommer' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Yoshi Tannamuri',
'ProductID': 'Gula Malacca', 'CustomerID': 'COMMI', 'CustomerName': 'Ann
Devon' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Palle Ibsen',
'ProductID': 'Rogede sild', 'CustomerID': 'RATTC', 'CustomerName': 'Paula
Wilson' },
        { 'OrderID': ++primaryKey, 'ProductName': 'Francisco Chang',
'ProductID': 'Mascarpone Fabioli', 'CustomerID': 'ROMEY', 'CustomerName':
'Jose Pavarotti' }
    ]
    let aData = (immutableGrid.dataSource as Object[]).concat(addedRecords);
    normalGrid.setProperties({ dataSource: aData });
    immutableGrid.setProperties({ dataSource: aData });
});
document.getElementById('reverse').addEventListener('click', function(e) {
    let aData = (immutableGrid.dataSource as Object[]).reverse();
    normalGrid.setProperties({ dataSource: aData });
    immutableGrid.setProperties({ dataSource: aData });
});
document.getElementById('paging').addEventListener('click', function(e) {
    let totalPage = (immutableGrid.dataSource as Object[]).length /
immutableGrid.pageSettings.pageSize;
    let page = Math.floor(Math.random() * totalPage) + 1;
    normalGrid.setProperties({ pageSettings: { currentPage: page } });
});

```

```

    immutableGrid.setProperties({ pageSettings: { currentPage: page } });
  });
  let normalGrid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    pageSettings: { pageSize: 50 },
    beforeDataBound: () => {
      normalStart = new Date().getTime();
    },
    dataBound: ()=> {
      let val = init ? '' : new Date().getTime() - normalStart;
      document.getElementById('normalDelete').innerHTML = 'Normal
      rendering Time: ' + "<b>" + val + "</b>" + '<b>ms</b>';
      normalStart = 0; init = false;
    },
    columns: [
      { field: 'OrderID', headerText: 'Order ID', isPrimaryKey: true,
      textAlign: 'Right', width: 120 },
      { field: 'ProductName', headerText: 'Product Name', width: 160 },
      { field: 'ProductID', headerText: 'Product ID', textAlign: 'Right',
      width: 120 },
      { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
      { field: 'CustomerName', headerText: 'Customer Name', width: 160 }
    ],
    height: 240
  });
  normalGrid.appendTo('#normal');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .immutablegrid,
    .normalgrid {
        pointer-events: none;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="template" type="text/x-template">
            <div class="image">
                
            </div>
        </script>
        <table>
            <tbody>
                <tr>
                    <td>
                        <span id="immutableDelete"></span>
                    </td>
                    <tr>
                        <td>
                            <span id="normalDelete"></span>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <div>
                                <button id="addtop" class="e-control e-btn e-lib
e-info">Add 5 rows at top</button>
                                <button id="addbottom" class="e-control e-btn e-
lib e-info">Add 5 rows at bottom</button>
                                <button id="delete" class="e-control e-btn e-lib
e-info">Delete 5 rows</button>
                                <button id="reverse" class="e-control e-btn e-
lib e-info">Sort Order ID</button>
                                <button id="paging" class="e-control e-btn e-lib
e-info">Paging</button>
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <span><b>Immutable Grid</b></span>
                            <div id="immutable" class="immutablegrid"></div>

```



```
        </td>
      </tr>
    <tr>
      <td>
        <span><b>Normal Grid</b></span>
        <div id="normal" class="normalgrid"></div>
      </td>
    </tr>
  </tbody>
</table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Limitations

The following features are not supported in the immutable mode:

- Frozen rows and columns
- Row Template
- Detail Template
- Hierarchy Grid
- Column reorder
- Virtual scroll
- Infinite scroll
- Grouping

Performance tips for EJ2 JavaScript Grid control

This article is a comprehensive guide on improving the loading performance of the EJ2 JavaScript Grid, especially when dealing with large datasets along with large number of columns. It provides valuable insights into the steps that need to be followed to bind a large data source without experiencing any performance degradations. By offering detailed explanations and actionable tips, this resource aims to empower readers with the knowledge and best practices necessary to optimize the performance of the EJ2 JavaScript Grid during data binding, ensuring a smooth and efficient user experience.

How to improve loading performance by binding large dataset

As you all know, a grid is made up of rows and columns. For instance, when you bind 10 rows and 10 columns, it means 100 elements will be rendered in the DOM (Document Object Model). So, it is recommended to render only a limited number of rows and columns to guarantee the best loading performance for the control.

Optimizing performance with paging

To boost the performance efficiency of your application, especially when dealing with large datasets, it is advised to implement paging. [Paging](#) allows you to display grid data in segmented pages, facilitating easier navigation through substantial datasets. This feature proves particularly beneficial in enhancing

the overall performance of your application. For more information on implementing paging, you can refer to the [documentation](#) section dedicated to this feature.

Optimizing performance with row virtualization or infinite scrolling

To enhance your application's efficiency, especially when dealing with substantial datasets, it is recommended to either using [virtualization](#) or [infinite scrolling](#). Implementing these techniques can significantly reduce the load on your application and elevate its overall performance.

1. **Virtualization:** The Virtual scrolling feature in the EJ2 JavaScript Grid enables the efficient handling and display of large volumes of data without compromising performance. This approach optimizes the rendering process by loading only the visible rows within the Grid viewport, rather than rendering the entire dataset simultaneously. For more information on implementing row virtualization , you can refer to the [documentation](#) section dedicated to this feature.
2. **Infinite scrolling:** The Infinite Scrolling feature in the EJ2 JavaScript Grid is a powerful tool for seamlessly handling extensive data sets without compromising grid performance. It operates on a "load-on-demand" concept, ensuring that data is fetched only when needed. In the default infinite scrolling mode, a new block of data is loaded each time the scrollbar reaches the end of the vertical scroller. For more information on implementing infinite scrolling , you can refer to the [documentation](#) section dedicated to this feature.

Optimizing performance with column virtualization in large no of columns

[Column virtualization](#) feature in the EJ2 JavaScript Grid that allows you to optimize the rendering of columns by displaying only the columns that are currently within the viewport. It allows horizontal scrolling to view additional columns. This feature is particularly useful when dealing with grids that have a large number of columns, as it helps to improve the performance and reduce the initial loading time.

It is possible to enable both row and column virtualization. This feature allows for efficient handling of large datasets by dynamically loading only the visible rows and columns, optimizing performance and enhancing the overall responsiveness of the grid. For more information on implementing column virtualization , you can refer to the [documentation](#) section dedicated to this feature.

How to overcome browser height limitation in virtual scrolling

[Documentation link](#)

How to improve loading performance by binding large data by showing custom text or element
When integrating image or template elements into a column, it's recommended to utilize the [Column Template](#) feature rather than customizing the data through [rowDataBound](#) or [queryCellInfo](#) events. These events are triggered for each row and cell rendering, introducing delays in the control's rendering process. Moreover, rendering custom elements using these events may result in the persistence of rendered elements, potentially causing longer rendering times over time. By opting for the column template feature, you can efficiently meet this requirement without experiencing rendering delays and ensure a more streamlined rendering process.

How to improve loading performance by referring individual script and CSS

To improve the performance of Syncfusion Grid control during the initial render as well as certain actions, suggested you to download the specific control scripts using CRG (Custom Resource Generator) to speed up the project. By default, the ej2.min.js script file contains all the Syncfusion control scripts. So, it will take some time to load the scripts to the project. Using [CRG](#), you can select the controls which

you want to use, and the modules for those controls, then you can download the scripts and CSS for the selected controls and use them as per your need.

[CRG website link](#)

So to improve the performance of grid during the initial rendering, suggested you to refer individual script and CSS.

[How to update cell values without frequent server calls](#)

Efficiently update cell values without the need for frequent server calls, especially beneficial for live update scenarios. Even when the data is initially bound from the server, performing edit operations can be done without triggering a database refresh. Utilize the [setCellValue](#) method to update the Grid without affecting the database and only refresh the UI.

[How to optimize server-side data operations with adaptors](#)

The EJ2 JavaScript Grid provides support for various adaptors (OData, ODataV4, WebAPI, URL, etc.) to facilitate server-side data operations and CRUD functionalities. By leveraging these adaptors along with the **DataManager** control, you can seamlessly bind remote data sources to the grid and execute actions. During data operations like filtering, sorting, and paging, the corresponding action queries are generated as per the adaptor's requirements. It is crucial to handle these actions on the application end and return the processed data back to the grid. Refer to the documentation for comprehensive details. It's worth noting that for efficient data processing, the suggested order for returning processed data to the grid is as follows

- Filtering
- Sorting
- Aggregates
- Paging
- Grouping

[How to avoid MaxJsonLength error while passing large amount of records](#)

The EJ2 JavaScript Grid control is client-server based. So, we send the data as JSON object between client and server. The reported issue occurs due to the serialization of the large-sized JSON object. We need to increase the maximum length for serializing the large-sized JSON object. You have to alter the [MaxJsonLength](#) property on your web.config file or in the place of deserialization.

Solution: 1

```
`csharp
<configuration>
<system.web.extensions>
<scripting>
<webServices>
<jsonSerialization maxJsonLength="25000000"/>
</webServices>
</scripting>
</system.web.extensions>
```

```
</configuration>
```

```
,
```

Solution : 2

```
`csharp
```

```
var serializer = new JavaScriptSerializer { MaxJsonLength = Int32.MaxValue };
,
```

Microsoft excel limitation while exporting millions of records to excel file format

By default, Microsoft Excel supports only 1,048,576 records in an excel sheet. Hence it is not possible to export millions of records to excel. You can refer the [documentation](#) link for more details on Microsoft excel specifications and limits. So suggest to export the data in CSV (Comma-Separated Values) or other formats that can handle large datasets more efficiently than Excel.

Columns

Columns in EJ2 JavaScript Grid control

The column definitions are used as the [dataSource](#) schema in the Grid. This plays a vital role in rendering column values in the required format. The grid operations such as sorting, filtering and grouping etc. are performed based on column definitions. The [field](#) property of the [columns](#) is necessary to map the data source values in Grid columns.

1. If the column [field](#) is not specified in the dataSource, the column values will be empty.

2. If the [field](#) name contains “dot” operator, it is considered as complex binding.

Column types

Column type can be specified using the [columns.type](#) property. It specifies the type of data the column binds.

If the [format](#) is defined for a column, the column uses [type](#) to select the appropriate format option ([number](#)

or [date](#)).

Grid column supports the following types:

- string
- number
- boolean
- date
- datetime

If the [type](#) is not defined, it will be determined from the first record of the [dataSource](#).

Incase if the first record of the [dataSource](#) is null/blank value for a column then it is necessary to define the [type](#) for that column.

ValueAccessor

The [valueAccessor](#) is used to access/manipulate the value of display data. You can achieve custom value formatting by using the [valueAccessor](#).

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', width: 100,
valueAccessor: currencyFormatter },
        { field: 'ShipName', headerText: 'Ship Name', width: 180,
valueAccessor: valueAccess }
    ],
    height: 315
});
grid.appendTo('#Grid');
function currencyFormatter(field: string, data: Object, column: Object):
string {
    return '€' + data['Freight'];
}
function valueAccess(field: string, data: Object, column: Object): string {
    return data[field] + '-' + data['ShipRegion'];
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Display array type columns

You can bind an array of objects in a column by using the [valueAccessor](#) property. In this example, the name field has an array of two objects, FirstName and LastName. These two objects are joined and bound to a column using the [valueAccessor](#).

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { stringData } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: stringData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 100 },
        { field: 'Name', headerText: 'Full Name', width: 120, valueAccessor:
valueAccess },
        { field: 'Title', headerText: 'Title', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');
function valueAccess(field: string, data: Object, column: Object): string {
    return data[field].map(function (s: {LastName: string, FirstName:
string}): string {
        return s.LastName || s.FirstName }).join(' ');
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Expression column

You can achieve the expression column by using the [valueAccessor](#) property.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { foodInformation } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: foodInformation,
    columns: [
        { field: 'FoodName', headerText: 'FoodName', width: 120 },
        { field: 'Protein', headerText: 'Protein', textAlign: 'Right',
width: 100 },
        { field: 'Fat', headerText: 'Fat', textAlign: 'Right', width: 100 },
        { field: 'Carbohydrate', headerText: 'Carbohydrate', textAlign:
'Right', width: 130 },
        { headerText: 'Calories In Take', valueAccessor: totalCalories,
textAlign: 'Right', width: 150 },
    ],
    height: 315
});
grid.appendTo('#Grid');
function totalCalories(field: string, data: { Protein: number, Fat: number,
Carbohydrate: number }, column: Object): number {
    return data.Protein * 4 + data.Fat * 9 + data.Carbohydrate * 4;
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Format

To format cell values based on specific culture, use the [columns.format](#) property. The grid uses [Internalization](#) library to format [number](#) and [date](#)

values.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120 },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID'
},
        { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign:
'Right', width: 140, format: 'yMd' }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the [number](#) and [date](#) values are formatted in **en-US** locale. You can localize the currency and date in different locale as explained [here](#)

Number formatting

The number or integer values can be formatted using the below format strings.

Format | Description | Remarks

{ type:'date', format:'dd/MM/yyyy' } | 04/07/1996

{ type:'date', format:'dd.MM.yyyy' } | 04.07.1996

{ type:'date', skeleton:'short' } | 7/4/96

{ type: 'dateTime', format: 'dd/MM/yyyy hh:mm a' } | 04/07/1996 12:00 AM

{ type: 'dateTime', format: 'MM/dd/yyyy hh:mm:ss a' } | 07/04/1996 12:00:00 AM

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'Freight', headerText: 'Freight', format: 'C2', width: 120
},
        { field: 'OrderDate', format: { type: 'date', format: 'dd/MM/yyyy' },
headerText: 'Order Date', width: 150 },
        { field: 'OrderDate', format: { type: 'dateTime', format: 'dd/MM/yyyy
hh:mm a' }, headerText: 'Ship Date', width: 180 }
    ],
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render boolean values as checkbox

To render boolean values as checkbox in columns, you need to set [displayAsCheckBox](#) property as **true**.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid(
{
    dataSource: data.slice(0,10),
    columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        {
            field: 'CustomerID', headerText: 'Customer ID', width: 150
        },
        { field: 'Freight', headerText: 'Freight', width: 100,
textAlign: 'Right' },
        { field: 'ShipName', headerText: 'Ship Name', width: 180 },
        { field: 'Verified', headerText: 'Verified', width: 150,
displayAsCheckBox: true }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
  </style>

```

```

    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Visibility

You can hide any particular column in Grid before rendering by defining [visible](#) property as false. In the below sample **ShipCity** column is defined as visible false.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'ShipCity', headerText: 'Ship City', width: 120 ,visible:
false},
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Lock columns

You can lock columns by using [column.lockColumn](#) property. The locked columns will be moved to the first position. Also you can't reorder its position.

In the below example, **Ship City** column is locked and its reordering functionality is disabled.

INDEX.TS

```

import { Grid, Reorder } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Reorder);
let grid: Grid = new Grid({
    dataSource: data,
    allowReordering: true,
    allowSelection: false,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100, lockColumn:
true, customAttributes: {class: 'customcss'}},
        { field: 'ShipName', headerText: 'Ship Name', width: 100 },
        { field: 'ShipPostalCode', headerText: 'Ship Postal code', width:
100 },
        { field: 'ShipRegion', headerText: 'Ship Region', width: 100 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="lock.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
#reorderMultipleCols {
    text-transform: none;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<script id="rowtemplate" type="text/x-template">
<tr>
<td class="photo">

</td>
<td class="details">
<table class="CardTable" cellpadding="3" cellspacing="2">
<colgroup>
<col width="50%">
<col width="50%">
</colgroup>
<tbody>
<tr>
<td class="CardHeader">First Name </td>
<td>${FirstName} </td>
</tr>
<tr>
<td class="CardHeader">Last Name</td>
<td>${LastName} </td>
</tr>

```

```

        <tr>
            <td class="CardHeader">Title
            </td>
            <td>${Title}
            </td>
        </tr>
        <tr>
            <td class="CardHeader">Country
            </td>
            <td>${Country}
            </td>
        </tr>
    </tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Controlling Grid actions

You can enable or disable grid action for a particular column by setting the [allowFiltering](#), [allowGrouping](#), [allowSorting](#), [allowReordering](#), and [allowEditing](#) properties.

INDEX.TS

```

import { Grid, Group, Sort, Filter, Edit, Reorder } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group, Sort, Filter, Edit, Reorder);
let grid: Grid = new Grid({
    dataSource: data,
    allowFiltering: true,
    allowGrouping: true,
    allowSorting: true,
    allowReordering: true,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Normal' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', allowGrouping: false,
        textAlign: 'Right', width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', allowFiltering: false,
        allowReordering: false, allowEditing: false, width: 100 },
    ]
});

```

```

        { field: 'ShipName', headerText: 'Ship Name', allowSorting: false,
width: 100 }
    ],
    height: 220
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide columns by external button

You can show or hide grid columns dynamically using external buttons by invoking the [showColumns](#) or [hideColumns](#) method.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 280
});
grid.appendTo('#Grid');
let show: Button = new Button({ cssClass: 'e-flat' }, '#show');
let hide: Button = new Button({ cssClass: 'e-flat' }, '#hide');
document.getElementById('show').onclick = () => {
    grid.showColumns(['Customer ID', 'Ship Name']); //show by HeaderText
};
document.getElementById('hide').onclick = () => {
    grid.hideColumns(['Customer ID', 'Ship Name']); //hide by HeaderText
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="show">Show</button>
        <button id="hide">Hide</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize column styles

You can customize the appearance of the header and content of a particular column using the [customAttributes](#) property.

To customize the grid column, follow the given steps:

Step 1:

Create a CSS class with custom style to override the default style for rowcell and headercell.

```

.e-grid .e-rowcell.customcss{
background-color: #ecedee;

```

```
color: 'red';
font-family: 'Bell MT';
font-size: 20px;
}
.e-grid .e-headercell.customcss{
background-color: #2382c3;
color: white;
font-family: 'Bell MT';
font-size: 20px;
}
,
```

Step 2:

Add the custom CSS class to the specified column by using the [customAttributes](#) property.

```
`ts
{ field: 'ShipCity', headerText: 'Ship City', customAttributes: {class: 'customcss'}, width: 100 },
,
```

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  enableHover: false,
  allowSelection: false,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', customAttributes: {
class: 'customcss' }, width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Display custom tooltip for columns

To display a custom Tooltip ([EJ2 Tooltip](#)), you can render the Grid control inside the Tooltip component and set the target as ".e-rowcell". The tooltip is displayed when hovering the grid cells.

Change the tooltip content for the grid cells by using the following code in the [beforeRender](#) event.

```
`ts
```

```
function beforeRender(args) {
```

// event triggered before render the tooltip on target element.

```
tooltip.content = args.target.closest("td").innerText;
}
,
```

INDEX.TS

```
import { Grid, QueryCellInfoEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { Tooltip } from '@syncfusion/ej2-popups';
let tooltip: Tooltip = new Tooltip({
    target: ".e-rowcell", // set the target element to show the tooltip on
    hovering it
    beforeRender: beforeRender
}, "#Tooltip");
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipName', headerText: 'Ship Name', width: 140 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    ],
    height: 315
});
grid.appendTo('#Grid');
function beforeRender(args) {
    // event triggered before render the tooltip on target element.
    tooltip.content = args.target.closest("td").innerText;
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Tooltip">
            <div id="Grid"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Align the text of Grid content and header

For aligning the text of Grid content and header part, kindly use [textAlign](#) and [headerTextAlign](#) properties.

Grid column supports the following alignments:

- Left
- Right
- Center
- Justify

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', type: 'number',
        textAlign: 'Right', headerTextAlign: 'Right', width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', type:
        'string', textAlign: 'Left', headerTextAlign: 'Left', width: 90 },
        { field: 'OrderDate', headerText: 'Order Date', type:
        'date', textAlign: 'Center', headerTextAlign: 'Center', format: 'yMd',
        width: 140 },
    ]
});

```

```

        { field: 'ShipCountry', headerText: 'Ship Country', type:
'string', textAlign: 'Justify', headerTextAlign: 'Justify', width: 120 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;

```

```

        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to prevent checkbox in the blank row

By default, cells in the grid will be blank if the corresponding column values in the data source are null or undefined. The grid also has the option to prevent the rendering of checkboxes in such cases, even if the [displayAsCheckBox](#) property is set to true for that column, by using the [rowDataBound](#) event of the Grid.

In the following sample, the `rowDataBound` event of the Grid is used to set the innerHTML of the checkbox element to empty.

INDEX.TS

```

import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    rowDataBound: rowDataBound,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'Freight', headerText: 'Freight', width: 100, textAlign:
'Right' },
        { field: 'ShipName', headerText: 'Ship Name', width: 180 },
        { field: 'Verified', headerText: 'Verified', width: 150,
displayAsCheckBox: true },
    ]
});
grid.appendTo('#Grid');
function rowDataBound(args) {
    let count = 0;
    let keys = Object.keys(args.data);
    for (let i = 0; i < keys.length; i++) {

```

```

        if (args.data[keys[i]] == null || args.data[keys[i]] == '' ||
args.data[keys[i]] == undefined) {
            count++;
        }
    }
    if (count == keys.length) {
        for (let i = 0; i < grid.columns.length; i++) {
            if (grid.columns[i].displayAsCheckBox) {
                args.row.children[i].innerHTML = '';
            }
        }
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [Group Column by Format](#)
- [How to set complex column as Foreignkey column](#)
- [Complex Data Binding with list of Array Of Objects](#)
- [Compare columns effortlessly using column pinning feature in Grid](#)
- [How to edit the column in Grid using the uploader control](#)
- [How to add the aggregate option functionalities in the column menu's feature](#)
- [How to render tooltip with custom content on Grid columns](#)
- [How to display the multiplication table of a number accepted from the user](#)
- [How to drag and drop within grid and between grids](#)
- [How to format date values received in JSON data through ajax post](#)

Auto generated columns in EJ2 JavaScript Grid control

The [columns](#) are automatically generated when [columns](#) declaration is empty or undefined while initializing the grid. All the columns in the [dataSource](#) are bound as grid columns.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
let data: Object[] = [
    { OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5 },
    { OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6 },
    { OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4 }];
let grid: Grid = new Grid ({
    dataSource: data
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title

```

```

                <td>${Title}
            </td>
        </tr>
        <tr>
            <td class="CardHeader">Country
            </td>
            <td>${Country}
            </td>
        </tr>
    </tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

When columns are auto-generated, the column [type](#) will be determined from the first record of the [dataSource](#).

Set isPrimaryKey for auto generated columns when editing is enabled

Primary key can be defined in the declaration of column object of the grid. When we didn't declare the columns, the grid will generate the columns automatically. For these auto generated columns, you can set [isPrimaryKey](#) column property as true by any one of the following two ways,

If Primary key "column index" is known then refer the following code example

`ts

```

import { Grid, Edit } from '@syncfusion/ej2-grids';
Grid.Inject(Edit);
let data: Object[] = [
    { OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5 },
    { OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6 },
    { OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4 }];
let grid: Grid = new Grid ({
    dataSource: data,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true }
});

```

```
grid.appendTo('#Grid');
grid.dataBound = () => {
  var column: Column = grid.columns[0];
  column.isPrimaryKey = 'true';
}
`ts
```

If Primary key "column fieldname" is known then refer the following code example

```
`ts
import { Grid, Edit } from '@syncfusion/ej2-grids';
Grid.Inject(Edit);
let data: Object[] = [
  { OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5 },
  { OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6 },
  { OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4 }];
let grid: Grid = new Grid ({
  dataSource: data,
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true }
});
grid.appendTo('#Grid');
grid.dataBound = () => {
  var column: Column = grid.getColumnByField('OrderID');
  column.isPrimaryKey = 'true';
}
`ts
```

Set column options to auto generated columns

You can set column options such as [format](#), [width](#) to the auto generated columns by using [dataBound](#) event of the grid.

In the below example, [width](#) is set for **OrderID** column, **date** type is set for **OrderDate** column and **numeric** type is set for **Freight** column.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
let data: Object[] = [
  { OrderID: 10248, CustomerID: 'VINET', Freight: 32.3800,
    OrderDate: "1996-07-02T00:00:00.000Z" },
  { OrderID: 10249, CustomerID: 'TOMSP', Freight: 32.3800,
    OrderDate: "1996-07-19T00:00:00.000Z" },
```



```

        { OrderID: 10250, CustomerID: 'HANAR', Freight: 32.3800,
OrderDate: "1996-07-22T00:00:00.000Z" }]];
let grid: Grid = new Grid ({
    dataSource: data,
    dataBound: dataBound
});
grid.appendTo('#Grid');
function dataBound(args: any): void {
    for (var i = 0; i < this.columns.length; i++) {
        this.columns[0].width = 120;
        if(this.columns[i].field === "OrderDate"){
            this.columns[i].type="date";
        }
        if (this.columns[i].type === "date") {
            this.columns[i].format = { type: "date", format:
"dd/MM/yyyy" };
        }
        this.columns[2].format = "P2";
    }
    this.refreshColumns();
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>${FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>${LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>${Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>${Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Headers in EJ2 JavaScript Grid control

Header text

By default, the header text of a column in Grid is displayed from the column's [field](#) value. However, you can easily override the default header title and provide a custom header text for the column using the [headerText](#) property.

To enable the `headerText` property, you simply need to define it in the [columns](#) property. The following example demonstrates how to enable header text for a Grid column.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
format: 'C', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
format: 'yMd', width: 140 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* The `headerText` property is optional, and if it is not defined, then the corresponding column's field value is set as header text for that column.

* You can also use the [headerTemplate](#) property to apply custom HTML content to the header cell.

Header template

The [headerTemplate](#) property is used to customize the header element of a Grid column. With this property, you can render custom HTML elements or EJ2 JavaScript control to the header element. This feature allows you to add more functionality to the header, such as sorting or filtering.

In this demo, the custom element is rendered for both **CustomerID** and **OrderDate** column headers.

INDEX.TS

```

import { Grid, ChangeEventArgs } from '@syncfusion/ej2-grids';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Switch } from '@syncfusion/ej2-buttons';
import { data } from './datasource.ts'
let dropdownData = ['Freight', 'Shipment', 'Cargo'];
let grid: Grid = new Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width:
120 },
    {
      field: 'CustomerID', headerText: 'Customer ID', width: 140,

```

```

        headerTemplate: ` <div> <span class="e-icon-userlogin e-icons
employee"></span> Customer ID</div>`,
    },
    { field: 'Freight', headerText: 'Freight', headerTemplate: ` <div
id='Freight'></div>`, width: 120 },
    {
        field: 'OrderDate', headerText: 'Order Date', format: 'yMd',
        headerTemplate:
            `<div><span id='OrderDate'></span>
<label id='headerText' style='margin-left:8px'>Order Date</label>
</div>`,
        width: 200,
    },
    ],
    height: 315
});
grid.appendTo('#Grid');
let dropDownColumn: DropDownList = new DropDownList({
    value: 'Freight',
    popupHeight: '200px',
    dataSource: dropdownData,
});
dropDownColumn.appendTo('#Freight');
let headerText = 'order date';
let toggleButton: Switch = new Switch({
    change: onSwitchToggle,
});
toggleButton.appendTo('#OrderDate');
function onSwitchToggle(args: ChangeEventArgs) {
    headerText = args.checked ? 'Purchase Date' : 'Order Date';
    (document.getElementById('headerText') as HTMLElement).innerHTML =
headerText;
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<style>
    @font-face {
        font-family: 'ej2-headertemplate';
        src:
            url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvTFIAAAEoAAAAVmNtYXDS2c5qAAABjAAAAEBnbHlmQmN
FrQAAAdQAAANoAGVhZBRdbkIAAADQAAAAANmhoZWEIUQOEAAAArAAAACRobXR4DAAAAAAAAAYAAAAA
MbG9jYQCOAbQAAAHMAAAACG1heHABHgENAAABCAAAACBuYW1lohGZJQAABTWAAAKPcG9zdA2o3w0
AAAFoAAAAQAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAwABAAAAQAATBXy9l8
PPPUACwQAAAAAANillKkAAAAA2KWUqQAAAAAD9APzAAAACAACAAAAAAAAAAAAEAAAADAQEAEQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAABQAAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLpYAQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAAAAAAAAgAAAAAMAAAAUAAAAQAAABQABAAsAAAAABgAEAAEAucC6WD//wAA5wLpYP//AAAAAA
BAAYABgAAAAIAAQAAAAAAjgG0ABEAAAAAA8kD8wADAAcACwAPABMAFwAbAB8AIwAnACsALwAzADc
AOwBPAGsAACUVIzUjFSM1IxUjNSMVizUjFSM1JRujNSMVizUjFSM1IxUjNSMVizU1FSM1IxUjNSM
VIzUjFSM1IxUjNQMVHwYhPwCRITcjDwghNS8HIzUjFSE1IwN2U1NTU1RTU1NTAUxTU1NTVFNTU1M
C7FNTU1NUU1NTU1QCAwUGBgIA0QICACHBQQBAVxsp30ICACHAQUDAQED1AECBAUHBwgIfVP+YFO
zU1NTU1NTU1NTU6dUVFVUFVFRUvUplNTU1NTU1NTU1P+NgQIBwcGBAMCAQIEBQcHAWgCdPoBAGQ
FAwCHCIF8CQgHBgUEAgFTU1MABAAAAAAD9APeADQABQCuAQAAAAEfCDc1Lwc1PwIPBy8HHww3HwQ
PATMVPwclLx0jDwMfAgUdAR8GBTUzLxQjDx0BFxUFEDsBPxA1Nyc1LxIPEhUCCg8OGxcVExANCgM
BAQMDCQQDAQgECAXESehMTEExUUFRTQTFBISEhEHCwYHCAKcKw0NDw8SuQQGAWIBAgRxlgsKCQCGBAM
BAGMDAwUFBQcGBWgICQkKCgsKDasMDQwNDQ4NDg45BQUDAQEEA/1eAgUGBwkKCwHjeggKDhEUFXs
1FRMSEA4NCwoJBwcJBjwODg0ODQ0MDQwLDAoLCgoJCQgIBwYHBQUFAwMDAgEBAQECAgYICg0ODxI
SFBUXFwwMDA0MDQwMFxcVFBISDw4MCwgGAgIBAQICAwJCw0PERITFRUXDAwMDA0NDawMDAsXFRQ
TEQ8ODQoIBgICAVQEAWgJCgsMCwwbEBAREREZE8VDAwKCGoIBwYFAwIBAQIDBQYHCAoUFQwLCws
LCgoJCACGMwsWfHUVHB3QAQIEBggICgueDg4ODg0NDA0MCwsLCwoKCQgJBwgGBwUFBAQDAwECCw8
NDxETCrIlawsKCAgGBAIB0AoLCwoLCQgNCAkLDA0NDg4ODg4ZFAlBAwMEBAUGBgYIBwkICQoKCws
LDAwMDA0ODQ4ODgH7DQwMDBgWFRQTERAODAoIBgICAQECAgYICgwOEBETFBWGAwMDA0MDQwMCxc
WFRMSEA8NDakHawIBAQEBAQECAwMICwwOEBETFBWFWwMDQAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAAQASAAEAQAAAAAAAAAgAHABMAAQAAAAAAAwASABoAAQAAAAAABAASACwAAQAAAAAABQALAD4
AAQAAAAAABgASAEKAAQAAAAAACgAsAFsAAQAAAAAACwASAIcAAwABBAKAAAAACAJkAAwABBAKAAQA
kAJsAAwABBAKAAgA0AL8AAwABBAKAAwAkAM0AAwABBAKABAakAPEAAwABBAKABQAWARUAAwABBAK
ABgAkASsAAwABBAKACgBYAU8AAwABBAKACwAkAacgZWoyLWhlYWRLcnRlbXBsYXRlUmVndWxhcmV
qMi1oZWZkZXJ0ZW1wbGF0ZWVqMi1oZWZkZXJ0ZW1wbGF0ZVZlcnNpb24gMS4wZWoyLWhlYWRLcnR
lbXBsYXRlRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Zlc2lubiBNZXRYbyBTdHVkaW93d3cuc3l
uY2Zlc2lubi5jb20AIAblAGoAMgAtAGgAZQBhAGQAZQByAHQAZQBtAHAAbABhAHQAZQBSAGUAZwB
1AGwAYQByAGUAagAyAC0AaABlAGEAZABlAHIAAdABlAG0AcABsAGEAdABlAGUAagAyAC0AaABlAGE
AZABlAHIAAdABlAG0AcABsAGEAdABlAFYAZQByAHMAaQBVAG4AIAAxAAC4AMABlAGoAMgAtAGgAZQ
hAGQAZQByAHQAZQBtAHAAbABhAHQAZQBGAG8AbgB0ACAAZwBlAG4AZQByAGEAdABlAGQAIABlAHM
AaQBuaGCAIAbTAHkAbgBjAGYAdQBzAGkAbwBuACAAATQB1AHQAcgBvACAAUwB0AHUAZABpAG8AdwB
3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAAAAAAAKAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAADAQIBAwEEAAtCVF9DYWxlbmRhcghlbXBsb3llZQAA)
format('truetype');
        font-weight: normal;
        font-style: normal;
    }
.e-icon-userlogin:before {

```

```

        font-family: 'ej2-headertemplate';
        content: "\e702";
    }
    .e-icon-userlogin {
        margin-right: 5px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* The `headerTemplate` property is only applicable to Grid columns that have a header element.

* You can use any HTML or EJ2 JavaScript control in the header template to add additional functionality to the header element.

Stacked header

In Grid, you can group multiple levels of column headers by stacking the Grid columns. This feature allows you to organize the Grid columns in a more structured and understandable way. This can be achieved by setting the `columns->columns` property. Within this property, you can define an array of column objects to group together as sub-headers under a main header. You can define the `headerText` property of each sub-header column to set the text for that sub-header.

You can customize the appearance of the stacked header elements by using the `headerTemplate` property. This property accepts an id reference, which allows you to define custom HTML elements or EJ2 JavaScript control to the header element. Here's an example of how to use stacked headers with a custom `headerTemplate` in Syncfusion Grid.

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,

    columns: [

```

```

        {field: 'OrderID',headerText: 'Order ID',width: 120,textAlign:
        'Center',headerTemplate: '#orderID',},
        {headerText: 'Order Details',headerTemplate: '#orderDate',
        columns: [
            {field: 'OrderDate',headerText: 'Order Date',textAlign:
            'Right',width: 135,format: 'yMd',minWidth: 10},
            {field: 'Freight',headerText: 'Freight($)',textAlign: 'Right',width:
            120,format: 'C2',minWidth: 10},
        ]
        },
        {headerText: 'Ship Details',
        headerTemplate: '<div> <span>Ship Details</span><span>(<i class="fa fa-
truck"></i></span></div>',
        columns: [
            {field: 'ShippedDate',headerText: 'Shipped Date',textAlign:
            'Right',width: 145,format: 'yMd',minWidth: 10},
            {field: 'ShipCountry',headerText: 'Ship Country',width:
            140,minWidth: 10},
        ],
        },
    ],
});
grid.appendTo('#Grid');
let dropdownData = ['Order Details', 'Order Information'];
let dropDownColumn: DropDownList = new DropDownList({
    value: 'Order Details',
    popupHeight: '200px',
    dataSource: dropdownData,
});
dropDownColumn.appendTo('#orderdate');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script id="orderId" let-data>
    <a href="#">OrderID</a>
  </script>
  <div id="orderDate" let-data>
    <div id="orderdate"></div>
  </div>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Align the text of header text

You can horizontally align the text in column headers of the Grid control using the [headerTextAlign](#) property. By default, the text is aligned to the left, but you can change the alignment by setting the value of the `headerTextAlign` property to one of the following options:

- **Left:** Aligns the text to the left (default).
- **Center:** Aligns the text to the center.
- **Right:** Aligns the text to the right.
- **Justify:** Header text is justified.

Here is an example of using the `headerTextAlign` property to align the text of a Grid column header:

INDEX.TS

```

import { Grid, ChangeEventArgs, ColumnModel } from '@syncfusion/ej2-grids';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { data } from './datasource.ts';
let grid: Grid = new Grid({

```

```

    dataSource: data,
    columns: [
      { field: 'OrderID', headerText: 'Order ID', width: 120 },
      { field: 'CustomerID', headerText: 'Customer Name', width: 150 },
      { field: 'Freight', headerText: 'Freight', format: 'C2', width: 120 },
      { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd' },
    ],
    height: 315
  });
grid.appendTo('#Grid');
let dropdownData = ['Left', 'Right', 'Center', 'Justify'];
let dropDownColumn: DropDownList = new DropDownList({
  value: 'Left',
  popupHeight: '240px',
  width: 100,
  dataSource: dropdownData,
  change: changeAlignment,
});
dropDownColumn.appendTo('#dropdown');
function changeAlignment(args: ChangeEventArgs) {
  grid.columns.forEach((column: ColumnModel) => {
    column.headerTextAlign = args.value
  })
  grid.refreshHeader();
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
  <div style="padding-bottom: 10px">
    <br>
    <label 30px 17px 0 0>Align the text of header text :</label>
    <input type="text" tabindex="1" id="dropdown" />
  </div>
  <div id="Grid">
  </div>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* The **headerTextAlign** property only changes the alignment of the text in the column header, and not the content of the column. If you want to align both the column header and content, you can use the [textAlign](#) property.

* You can also use the **headerTextAlign** property with the stacked header feature in Syncfusion Grid. The property will align the header text in the sub-headers as well.

[Autowrap the header text](#)

The autowrap allows the cell content of the grid to wrap to the next line when it exceeds the boundary of the specified cell width. The cell content wrapping works based on the position of white space between words. To support the Autowrap functionality in Syncfusion Grid, you should set the appropriate [width](#) for the columns. The column width defines the maximum width of a column and helps to wrap the content automatically.

To enable autowrap, set the **allowTextWrap** property to **true**. You can also configure the auto wrap mode by setting the [textWrapSettings.wrapMode](#) property.

Grid provides the below three options for configuring:

- **Both:** This is the default value for wrapMode. With this option, both the grid header text and content is wrapped.
- **Header:** With this option, only the grid header text is wrapped.
- **Content:** With this option, only the grid content is wrapped.

* If a column width is not specified, then the Autowrap of columns will be adjusted with respect to the grid's width.

* If a column's header text contains no white space, the text may not be wrapped.

* If the content of a cell contains HTML tags, the Autowrap functionality may not work as expected. In such cases, you can use the [headerTemplate](#) and [template](#) properties of the column to customize the appearance of the header and cell content.

In the example below, the `textWrapSettings.wrapMode` property is set to **Header** only the grid header text is wrap to the next line.

INDEX.TS

```
import { Grid, Page, ChangeEventArgs } from '@syncfusion/ej2-grids';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { inventoryData } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid(
    {
        dataSource: inventoryData,
        allowPaging: true,
        allowTextWrap: true,
        textWrapSettings: { wrapMode: 'Header' },
        columns: [
            { field: 'Inventor', headerText: 'Inventor Name', width: 180,
textAlign: 'Right' },
            { field: 'NumberofPatentFamilies', headerText: 'Number of Patent
Families', width: 180 },
            { field: 'Country', headerText: 'Country', width: 140 },
            { field: 'Active', headerText: 'Active', width: 120 },
            { field: 'Mainfieldsofinvention', headerText: 'Main Feilds Of
Invention', width: 200 },
        ],
    });
grid.appendTo('#Grid');
let dropdownData = [
    { text: 'Header', value: 'Header' },
    { text: 'Both', value: 'Both' },
]
let dropDownColumn: DropDownList = new DropDownList({
    dataSource: dropdownData,
    popupHeight: '240px',
    width: '120px',
    value: 'Header',
    change: change,
});
dropDownColumn.appendTo('#dropdownlist');
function change(args: ChangeEventArgs) {
    grid.textWrapSettings.wrapMode = args.value;
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div style="padding-bottom: 10px">
    <label padding: 30px 17px 0 0>Autowrap for header column :</label>
    <input type="text" tabindex="1" id="dropdownlist" />
  </div>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Change the height of header

Changing the height of the header can be useful in cases where the default height is not sufficient to display the header content cell. For example, if you have a header with a lot of text or if you want to add an image to the header, you may need to increase the height of the header to accommodate the content. This can be easily achieved by changing the height of the header using CSS or by dynamically adjusting the height using a methods.

Using CSS

You can use CSS to override the default height of the **.e-grid .e-headercell** class to change the height of the header. Here is an example code snippet:

```
`css
```

```
.e-grid .e-headercell {
height: 130px;
}
,
```

Using methods

To change the height of the header dynamically, you can use the [getHeaderContent](#) method to get the header content element of the Syncfusion Grid. Then, you can use the **querySelectorAll** method to get all the header cell elements with the class **e-headercell**. Finally, you can loop through each header cell element and set its style property to adjust the height.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 140 },
        { field: 'Freight', headerText: 'Freight', format: 'C2', width: 120 },
        { field: 'OrderDate', headerText: 'Order Date', format: 'yMd',
width: 140 },
    ]
});
grid.appendTo('#Grid');
let button: Button = new Button(
    {
        content: 'CHANGE HEIGHT TO 20PX',
    }
)
button.appendTo('#small');
let button1: Button = new Button(
    {
        content: 'DEFAULT HEIGHT TO 42PX',
    }
);
button1.appendTo('#medium');
let button3: Button = new Button(
    {
        content: 'CHANGE HEIGHT TO 60PX',
    }
);
button3.appendTo('#big');
(<HTMLElement>document.getElementById('changeHeight')).onclick = function
(event) {
    let heightMap = { small: '20px', medium: '42px', big: '60px' };
    let headerCells = (grid).getHeaderContent().querySelectorAll('.e-
headercell');
    headerCells.forEach((headerCell: HTMLElement) => {
        (headerCell as HTMLElement).style.height = (heightMap)[
            (event.target as HTMLButtonElement).id
        ];
    });
};
```

```

    });
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="changeHeight">
    <button ej-button id="small" cssClass="e-small">
      Change height to 20px
    </button>
    <button ej-button id="medium" cssClass="e-small">
      Default height to 40px
    </button>
    <button ej-button id="big" cssClass="e-small">
      Change height to 60px
    </button>
  </div>
  <br>
  <div id="container">
    <div id="Grid"></div>
  </div>

```

```

<script>
    var ele = document.getElementById('container');
    if (ele) {
        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* You can also use the [getHeaderTable](#) method to get the table element of the header, and then adjust the height.

* You cannot change the height of row below the default height of 42px using the **e-columnheader** class.

Change the header text dynamically

The Syncfusion Grid controls provides a way to modify the header text of a corresponding column in real-time based on events or other events. This feature can be useful in various scenarios, such as displaying a custom header text for a specific column or updating the header text dynamically based on input. By allowing for dynamic changes to the header text, the Grid provides a more flexible and customizable experience.

Using event

To modify the header text of a corresponding column dynamically, you can use the [headerCellInfo](#) event provided by the Syncfusion Grid. This event is triggered for each header cell element rendered in the Grid.

When the `headerCellInfo` event is triggered, it provides a **HeaderCellInfoEventArgs** object as a parameter. This object contains the following properties:

- **cell**: Defines the header cell that is being modified.
- **node**: Defines the DOM element of the header cell that is being modified.

You can use these properties to access and modify the header text of the corresponding column. Once the header text is modified, you can refresh the Grid to reflect the changes by calling the [refreshHeader](#) method of the Grid.

Using method

The Grid control provides several methods that allow you to change the column header text dynamically. Here are some of the methods you can use:

1. [getColumnByField](#): This method takes a field name as a parameter and returns the entire column object that corresponds to that field name, including properties such as headerText, width, and alignment. You can use this method to modify any aspect of the column. 2. [getColumnHeaderByField](#): Retrieves the header element of a column based on its field name. You can modify the **textContent** property of the header element to change the header text. This method does not return a reference to the column object itself, only to the header element. 3. [getColumnIndexByField](#): Retrieves the index of a column based on its field name. You can then use the `getColumnByIndex` method to retrieve the column object and modify its `headerText` property to change the header text. 4. [getColumnByUid](#): Retrieves the column object based on its unique identifier (UID). You can modify the `headerText`

property of the column object to change the header text. 5. [getColumnHeaderByIndex](#): Retrieves the header element of a column based on its zero-based index. You can modify the **textContent** property of the header element to change the header text. This method does not return a reference to the column object itself, only to the header element. 6. [getColumnIndexByUid](#): Retrieves the index of a column based on its unique identifier (UID). You can then use the [getColumnByIndex](#) method to retrieve the column object and modify its **headerText** property to change the header text. 7. [getColumnHeaderByUid](#): Retrieves the header element of a column based on its unique identifier (UID). You can modify the **textContent** property of the header element to change the header text. This method does not return a reference to the column object itself, only to the header element. If you only have an **template** for the column header, and the column itself is not defined with a **field**, then you can use the [getColumnHeaderByUid](#) method to get a reference to the header element and modify its text content to change the header text.

* When you change the header text dynamically, make sure to **refresh** the Grid to reflect the changes by calling the [refreshHeader](#) method.

* The UID is automatically generated by the Grid control and may change whenever the grid is refreshed or updated.

Here is an example of how to change the header text of a column using the [getColumnByField](#) method:

INDEX.TS

```
import { Grid, Page } from '@syncfusion/ej2-grids';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { data } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
import { TextBox } from '@syncfusion/ej2-inputs';
Grid.Inject(Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'Freight', width: 120, format: 'C2' },
    { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd' },
  ]
});
grid.appendTo('#Grid');
let columns =
[
  { text: 'OrderID', value: 'OrderID' },
  { text: 'CustomerID', value: 'CustomerID' },
  { text: 'Freight', value: 'Freight' },
  { text: 'OrderDate', value: 'OrderDate' },
]
let field = { text: 'text', value: 'value' };
let dropDownColumn: DropDownList = new DropDownList({
  dataSource: columns,
  fields: field,
  value: 'OrderID',
  popupHeight: '240px',
  width: '120px',
```

```

});
dropDownColumn.appendTo('#dropdownlist');
let textbox: TextBox = new TextBox({
    placeholder: 'Enter new header text',
    width: 140,
});
textbox.appendTo('#textboxvalue');
let button: Button = new Button({
    content: 'Change',
});
button.appendTo('#buttons');
(document.getElementById('buttons') as HTMLElement).onclick = function () {
    if (textbox.value.trim() !== '') {
        let column = grid.getColumnByField(dropDownColumn.value);
        column.headerText = textbox.value;
        grid.refreshHeader();
    }
};

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>

```

```

<body>
  <div id="container">
    <div style="padding-bottom: 10px">
      <label padding: 30px 17px 0 0>Select a column Name : </label>
      <input type="text" tabindex="2" id="dropdownlist" />
    </div>
    <div style="padding-bottom: 10px">
      <label padding: 30px 17px 0 0>Enter New Header Text : </label>
      <input type="text" tabindex="2" id="textboxvalue" />
    </div>
    <div style="padding-bottom: 10px">
      <label padding: 30px 17px 0 0>Click the change button : </label>
      <button ej-button id="buttons" cssClass="e-small"></button>
    </div>
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Changing header text using headerValueAccessor property

The [headerValueAccessor](#) property in Syncfusion Grid allows you to customize the text of a column header cell, which can be useful in scenarios where you want to change the text to display it in a different language, format or add additional information to the header. This property is triggered every time the header cell is rendered.

To enable the [headerValueAccessor](#) property, you need to set the [headerValueAccessor](#) property of the corresponding column. This property accepts a callback function that takes two arguments:

- **field:** Represents the current field of the column.
- **column:** Represents the current column object.

* The [headerValueAccessor](#) property should only be used to change the text of the header and not to perform any DOM-oriented operations such as adding or manipulating DOM elements in the header. In such cases, you should use the [headerCellInfo](#) event instead.

* The [headerValueAccessor](#) property is triggered every time the header cell is rendered or refreshed.

* The callback function defined for the [headerValueAccessor](#) property should return a string that represents the new text of the column header.

* If you only need to refresh the column header, you can dynamically change the header content using the [refreshHeader](#) method.

* You can use this property for individual columns or for all columns by adding it to the grid's properties.

Here's an example of how to use the `headerValueAccessor` property to change the header text of a column:

INDEX.TS

```
import { Grid, Page, ColumnModel } from '@syncfusion/ej2-grids';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { data } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
import { TextBox } from '@syncfusion/ej2-inputs';
Grid.Inject(Page);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120,
headerValueAccessor: headerValueAccessor },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150,
headerValueAccessor: headerValueAccessor },
        { field: 'Freight', width: 120, format: 'C2', headerValueAccessor:
headerValueAccessor },
        { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', headerValueAccessor: headerValueAccessor },
    ]
});
grid.appendTo('#Grid');
let columns = [
    { text: 'OrderID', value: 'OrderID' },
    { text: 'CustomerID', value: 'CustomerID' },
    { text: 'Freight', value: 'Freight' },
    { text: 'OrderDate', value: 'OrderDate' },
];
let field = { text: 'text', value: 'value' };
let dropDownColumn: DropDownList = new DropDownList({
    dataSource: columns,
    fields: field,
    value: 'OrderID',
    popupHeight: '240px',
    width: '120px',
});
dropDownColumn.appendTo('#dropdownlist');
let textbox: TextBox = new TextBox({
    placeholder: 'Enter new header text',
    width: 140,
});
textbox.appendTo('#textboxvalue');
let button: Button = new Button({
    content: 'Change',
});
button.appendTo('#buttons');
function headerValueAccessor(field, columns: ColumnModel) {
    if (textbox && textbox.value && textbox.value.trim() !== '' &&
columns.field === dropDownColumn.value) {
        columns.headerText = textbox.value;
    }
}
(<HTMLElement>document.getElementById('buttons')).onclick = function () {
```

```
grid.refreshHeader();
};
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div style="padding-bottom: 10px">
      <label padding: 30px 17px 0 0>Select a column Name : </label>
      <input type="text" tabindex="2" id="dropdownlist" />
    </div>
    <div style="padding-bottom: 10px">
      <label padding: 30px 17px 0 0>Enter New Header Text : </label>
      <input type="text" tabindex="2" id="textboxvalue" />
    </div>
    <div style="padding-bottom: 10px">
      <label padding: 30px 17px 0 0>Click the change button : </label>
      <button ej2-button id="buttons" cssClass="e-small"></button>
    </div>
    <div id="Grid"></div>
  </div>
```

```

<script>
  var ele = document.getElementById('container');
  if (ele) {
    ele.style.visibility = "visible";
  }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Changing the header text of all columns

If you want to change the header text of all columns in the grid, you can loop through the Columns collection of the grid and set the `headerText` property for each column. Here is an example:

INDEX.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { Grid, ColumnModel } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  height: 280,
  columns: [
    { field: 'OrderID', headerText: 'OrderID', textAlign: 'Right', width:
100 },
    { field: 'CustomerID', headerText: 'CustomerID', width: 120 },
    { field: 'Freight', headerText: 'Freight', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', format: 'yMd', width: 100
  },
  ],
});
grid.appendTo('#Grid');
let headerTextMap = {
  'OrderID': 'Order ID',
  'CustomerID': 'Customer ID',
  'Freight': 'Freight Charge',
  'ShipCity': 'Ship To City',
}
let button: Button = new Button({
  content: 'Change Header Text',
  cssClass: "e-success",
});
button.appendTo('#buttons');
(document.getElementById('buttons') as HTMLElement).onclick = function () {
  grid.columns.forEach((column: ColumnModel) => {
    column.headerText = headerTextMap[column.field];
  });
  grid.refreshHeader();
};

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>

```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <button ej2-button id="buttons" cssClass="e-success"></button>
    <br><br>
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Change the orientation of header text

By default, the text in the column headers of the Syncfusion Grid controls is oriented horizontally. However, in some cases, you may want to change the orientation of the header text to vertical, diagonal, or at a custom angle. This can be achieved by adding a custom CSS class to the column header cell using the [customAttributes](#) property of the Grid columns.

Follow the below steps to change the orientation of the header text in Grid:

Step 1: Create a CSS class with orientation style for grid header cell

To **rotate** the header text, you can create a CSS class with the **transform** property that rotates the header text 90 degrees. This class will be added to the header cell using the **customAttributes** property.

```
`css
.orientationcss .e-headercelldiv {
transform: rotate(90deg);
}
`
```

Step 2: Add the custom CSS class to the grid column

Once you have created the CSS class, you can add it to the particular column by using the **customAttributes** property. This property allows you to add any custom attribute to the grid column.

For example, to add the orientationcss class to the Freight column, you can use the following code:

```
`typescript
{ field: 'Freight', headerText: 'Freight', textAlign: 'Center', format: 'C2', customAttributes: {class:
'orientationcss'}, width: 80}
`
```

Step 3: Resize the header cell height

After adding the custom CSS class to a column, you need to resize the header cell height so that the rotated header text is fully visible. You can do this by using the following code:

```
`typescript
setHeaderHeight(args) {
let textWidth: number = document.querySelector('.orientationcss > div').scrollWidth;//Obtain the width
of the headerText content.
let headerCell: NodeList = document.querySelectorAll('.e-headercell');
for(let i: number = 0; i < headerCell.length; i++) {
(<HTMLElement>headerCell.item(i)).style.height = textWidth + 'px'; //Assign the obtained textWidth as
the height of the headerCell.
}
}
`
```

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let customAttributes = { class: 'orientationcss' };
let grid: Grid = new Grid({
  dataSource: data,
```



```

created: setHeaderHeight,
height: 240,
columns: [
  { field: 'OrderID', headerText: 'Order ID', width: 100 },
  { field: 'CustomerID', headerText: 'Customer Name', width: 120 },
  { field: 'Freight', width: 80, customAttributes: customAttributes,
format: 'C2', textAlign: 'Center' },
  { field: 'ShipCity', headerText: 'Order Date', width: 100, format: 'yMd'
},
]
});
customAttributes = { class: 'orientationcss' };
grid.appendTo('#Grid');
function setHeaderHeight() {
  let textWidth = (<HTMLElement>document.querySelector('.orientationcss >
div')).scrollWidth;
  //Obtain the width of the headerText content.
  let headerCell = document.querySelectorAll('.e-headercell');
  for (let i = 0; i < headerCell.length; i++) {
    (<HTMLElement>headerCell.item(i)).style.height = textWidth + 'px';
  }
  //Assign the obtained textWidth as the height of the headerCell.
}
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<style>
  body {
    touch-action: none;
  }
  .orientationcss .e-headercelldiv {
    transform: rotate(90deg);
  }
</style>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Custom tooltip for header

Custom tooltips for headers provide additional information when hovering over a column header in the Syncfusion Grid. This can be useful in situations where there is not enough space to display all of the information related to a column, or when there is additional context that may be helpful.

To enable custom tooltips for headers, you can use the [beforeRender](#) event of the Grid control. This event is triggered for each header cell before it is rendered, allowing you to add a custom tooltip to the header cell using [tooltip](#) control.

Here's an example of how to use the `beforeRender` event to add a custom tooltip to a header cell:

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { Tooltip, TooltipEventArgs } from '@syncfusion/ej2-popups';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width:
120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right', width:
120, format: 'C2' },
    { field: 'ShipName', headerText: 'Ship Name', textAlign: 'Right', width:
145, format: 'yMd' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 140 },

```

```

    ],
  });
  grid.appendTo('#Grid');
  let columnDescriptions: Object = {
    'Order ID': 'A unique number assigned to each order.',
    'Freight': 'The cost of shipping the order.',
    'Ship Name': 'The name of the person or company who will receive the shipment.',
    'Ship Country': 'The country to which the shipment will be sent.',
    'Order Date': 'The date when the order was placed.',
  };
  let tooltip = new Tooltip({
    beforeRender: beforeRender,
    target: '.e-headertext',
  });
  tooltip.appendTo('#tooltip');
  function beforeRender(args: TooltipEventArgs) {
    let description = (columnDescriptions as Object)[args.target.innerText];
    if (description) {
      (tooltip as Tooltip).content =
        args.target.innerText + ': ' + description;
    }
  }
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="tooltip">
      <div id="Grid"></div>
    </div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* The [headerCellInfo](#) event can also be used to customize the header tooltip. This event is triggered for each header cell after it is rendered.

Customize header text styles

Customizing the grid header styles allows you to modify the appearance of the column header in the Grid control to meet your design requirements. You can customize the font, background color, and other styles of the header cells. To customize the header styles in the grid, you can use CSS, properties, methods, or event support provided by the Syncfusion EJ2 JavaScript Grid control.

Using CSS

You can apply styles to the header cells using CSS selectors. The Grid provides a class name for each header cell element, which you can use to apply styles to that specific header cell. The **.e-headercell** class can be used to change the background color and text color of the column header.

`CSS

```

.e-grid .e-headercell {
background-color: #a2d6f4;
color:rgb(3, 2, 2);
}
`

```

Here's an example that demonstrates how to customize the appearance of a specific column header in the Grid using **className**.

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({

```

```

dataSource: data,
allowPaging: true,
columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right' },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150, textAlign:
'Right' },
    { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
    { field: 'Freight', headerText: 'Freight', width: 140, format: 'C2',
textAlign: 'Right' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 130, format:
'yMd', textAlign: 'Right' },
]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <style>
        .e-grid .e-headercell {
            background-color: #a2d6f4;
            color: rgb(3, 2, 2);
        }
    </style>

```

```

</style>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Using property

You can customize the appearance of the column headers in Grid using the [customAttributes](#) property. The `customAttributes` property takes an object with the name-value pair to customize the CSS properties for grid header cells. You can also set multiple CSS properties to the custom class using the `customAttributes` property.

To customize the header of a column, you can follow the steps below:

Step 1: Define a CSS class that specifies the styles you want to apply to the header cell of the column. For example, to change the background color and text color of the header cell, define a CSS class like this:

```

`CSS
.e-grid .e-headercell.customcss {
background-color: rgb(43, 205, 226);
color: black;
}
`

```

Step 2: Set the `customAttributes` property of the desired column to an object that contains the CSS class **customcss**. This CSS class will be applied to the header cell of the specified column in the Grid.

```

`ts
{field: "Freight" headerText: "Freight" customAttributes: {class: '.customcss'}}
`

```

The following example demonstrates how to customize the appearance of the **OrderID** and **Freight** columns using custom attributes.

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';

```

```

Grid.Inject(Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  columns: [
    {field: 'OrderID', headerText: 'Order ID', customAttributes: { class:
'customcss' }, textAlign: 'Center'},
    { field: 'CustomerName', headerText: 'Customer Name', textAlign:
'Center'},
    {field: 'Freight', headerText: 'Freight', customAttributes: { class:
'customcss' }, textAlign: 'Center'},
    {field: 'OrderDate', headerText: 'Order Date', format: 'yMd', textAlign:
'Center'}
  ],
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <style>
    .e-grid .e-headercell.customcss {
      background-color: rgb(43, 205, 226);
      color: black;
    }
  </style>

```

```

</style>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Using method

The Syncfusion Grid provides methods to customize the appearance of the grid columns header.

1. [getColumnHeaderByIndex](#): The method is used to customize the appearance of a specific column header in the grid by specifying the index of the column for which you want to customize the header.
2. [getColumnHeaderByField](#): This method is used to retrieve the header element of a specific column by its field name. You can use the retrieved element to customize the appearance of the header element.
3. [getColumnHeaderByUid](#): This method is used to retrieve the header element of a specific column by its unique ID. You can use the retrieved element to customize the appearance of the header element.
4. [getColumnIndexByField](#): This method is used to retrieve the index of a specific column by its field name. You can use the retrieved index to access the header element and customize its appearance.
5. [getColumnIndexByUid](#): This method is used to retrieve the index of a specific column by its unique ID. You can use the retrieved index to access the header element and customize its appearance.

Here's an example of how to use these methods to change the style of a specific column header:

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  dataBound: dataBound,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width:
120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 140 },

```



```

    { field: 'Freight', headerText: 'Freight($)', textAlign: 'Right', width:
120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 140 },
  ],
});
grid.appendTo('#Grid');
function dataBound() {
  (grid.getColumnHeaderByIndex(0) as HTMLElement).style.color = '#0d0b0b';
  (grid.getColumnHeaderByField('CustomerID') as
HTMLElement).style.background = '#f45ddeab';
  (grid.getColumnHeaderByField('CustomerID') as HTMLElement).style.color =
'#0d0b0b';
  (grid.getColumnHeaderByUid('grid-column2') as
HTMLElement).style.background = '#f45ddeab';
  let columnIndex = grid.getColumnIndexByField('ShipCountry');
  (grid.getColumnHeaderByIndex(columnIndex) as HTMLElement).style.color =
'#0d0b0b';
  let index = grid.getColumnIndexByUid('grid-column2');
  (grid.getColumnHeaderByIndex(index) as HTMLElement).style.color =
'#0d0b0b';
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>

```

```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

* The UID is automatically generated by the Grid control and may change whenever the grid is refreshed or updated.

Using event

To customize the appearance of the grid header, you can handle the [headerCellInfo](#) event of the grid. This event is triggered when each header cell is rendered in the grid, and provides an object that contains information about the header cell. You can use this object to modify the styles of the header column.

The following example demonstrates how to add a `headerCellInfo` event handler to the grid. In the event handler, checked whether the current header column is **Order Date** field and then applied the appropriate CSS class to the cell based on its value.

INDEX.TS

```
import { Grid, Page, HeaderCellInfoEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid(
  {
    dataSource: data,
    allowPaging: true,
    headerCellInfo: onHeaderCellInfo,
    columns: [
      { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120 },
      { field: 'CustomerID', headerText: 'Customer ID', textAlign:
'Right', width: 120 },
      { field: 'OrderDate', headerText: 'Order Date', textAlign:
'Right', width: 135, format: 'yMd' },
      { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C2' },
      { field: 'ShippedDate', headerText: 'Shipped Date',
textAlign: 'Right', width: 145, format: 'yMd' },
    ]
  });
grid.appendTo('#Grid');
function onHeaderCellInfo(args: HeaderCellInfoEventArgs)
```

```
{
    if (args.cell.column.field == 'OrderDate') {
        args.node.classList.add('customcss');
    }
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <style>
        .e-grid .e-headercell.customcss {
            background-color: rgb(43, 205, 226);
            color: black;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
```

```

        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

How to refresh header

The refresh header feature in the Syncfusion EJ2 JavaScript Grid allows you to update the header section of the grid whenever changes are made to the grid's columns. This feature is useful when you want to reflect changes in the header immediately, such as modifying the column header text, width, or alignment.

To use the refresh header feature, you can call the [refreshHeader](#) method of the Grid control. This method updates the grid header with the latest changes made to the columns.

The following example demonstrates how to use the `refreshHeader` method to update the grid header:

INDEX.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width:
120 },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 135, format: 'yMd' },
        { field: 'Freight', headerText: 'Freight($)', textAlign: 'Right', width:
120, format: 'C2' },
        { field: 'ShipCity', headerText: 'Ship City', textAlign: 'Right', width:
100 },
    ],
});
grid.appendTo('#Grid');
let button = new Button({
    content: 'Refresh Header',
});
button.appendTo('#buttons');
(<HTMLElement>document.getElementById('buttons')).onclick = function () {
    let column = grid.getColumnByIndex(1);
    column.headerText = 'New Header Text'; // update the header text of the
column object
    grid.refreshHeader();
};

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div style="padding-bottom: 10px">
      <button ej-button id="buttons" cssClass="e-small"></button>
      <br><br>
      <div id="Grid"></div>
    </div>
    <script>
      var ele = document.getElementById('container');
      if (ele) {
        ele.style.visibility = "visible";
      }
    </script>
    <script src="index.js" type="text/javascript"></script>
  </body>
</html>

```

* The `refreshHeader` method updates only the grid header and not the entire grid.

* If you want to refresh the entire grid, you can use the `refresh` method instead.

[How to get header element](#)

To get the header element in a Syncfusion Grid, you can use one of the following methods:

1. [getHeaderContent](#): This method returns the header div element of the Grid. You can use this method to access the entire header content of the Grid.

```
`ts
```

```
const headerElement = grid.getHeaderContent();
```

```
,
```

2. [getHeaderTable](#): This method returns the header table element of the Grid. You can use this method to access only the header table of the Grid.

```
`ts
```

```
const headerTableElement = grid.getHeaderTable();
```

```
,
```

3. [getColumnHeaderByUid](#): This method returns the column header element by its unique identifier.

```
`ts
```

```
const columnHeaderElement = grid.getColumnHeaderByUid("e-grid2");
```

```
,
```

4. [getColumnHeaderByIndex](#): This method returns the column header element by its index.

```
`ts
```

```
const columnHeaderElement = grid.getColumnHeaderByIndex(0);
```

```
,
```

5. [getColumnHeaderByField](#): This method returns the column header element by its field name.

```
`ts
```

```
const columnHeaderElement = grid.getColumnHeaderByField("OrderID");
```

```
,
```

* The UID is automatically generated by the Grid control and may change whenever the grid is refreshed or updated.

Column template in EJ2 JavaScript Grid control

Render image in a column

The column [template](#) has options to display custom element instead of a field value in the column.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { employeeData } from '../datasource.ts';
let grid: Grid = new Grid({
  dataSource: employeeData,
  columns: [
    {
      headerText: 'Employee Image', textAlign: 'Center',
      template: '#template', width: 150
    },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 125 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'Title', headerText: 'Title', width: 170 }
  ],
},
```

```

        height: 315
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="template" type="text/x-template">
            <div class="image">
                
            </div>
        </script>
        <div id="Grid"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Grid actions such as editing, grouping, filtering and sorting etc. will depend upon the column [field](#). If the [field](#) is not specified in the template column, the grid actions cannot be performed.

Render other components in a column

You can render any component in a grid column using the [template](#) property.

To render other components in the grid, ensure the following steps:

Step 1:

Initialize the column template for your custom component.

```
`ts
template:`<div>
<select class="e-control e-dropdownlist">
<option value="1" selected="selected">Order Placed</option>
<option value="2">Processing</option>
<option value="3">Delivered</option>
</select>
</div>`
`
```

Step 2:

Using the [queryCellInfo](#) event, you can render the DropDown component with the following code.

```
`ts
function dropdown(args: QueryCellInfoEventArgs) {
let ele=args.cell.querySelector('select');
let drop = new DropDownList({popupHeight: 150, popupWidth: 150});
drop.appendTo(ele);
}
`
```

INDEX.TS

```
import { Grid, QueryCellInfoEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
```



```

let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        {
            headerText: 'Order Status',
            template:
                `<div>
                    <select class="e-control e-dropdownlist">
                        <option value="1" selected="selected">Order
Placed</option>
                        <option value="2">Processing</option>
                        <option value="3">Delivered</option>
                    </select>
                </div>`, width: 140
        },
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'OrderDate', headerText: 'Order Date', width: 100, format:
'yMd' },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    ],
    height: 315,
    queryCellInfo: dropdown
});
grid.appendTo('#Grid');
function dropdown(args: QueryCellInfoEventArgs): void {
    let ele: HTMLSelectElement = args.cell.querySelector('select');
    let drop: DropDownList = new DropDownList({ popupHeight: 150,
popupWidth: 150 });
    drop.appendTo(ele);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using condition template

You can render the template elements based on condition.

In the following code, checkbox is rendered based on **Discontinued** field value.

```

<script id="template" type="text/x-template">
<div class="template_checkbox">
    ${if(Discontinued)}
    <input type="checkbox" checked> ${else}
    <input type="checkbox"> ${/if}
</div>
</script>

```

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
```

```
import { productData } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: productData,
    columns: [
        {
            headerText: 'Discontinued', textAlign: 'Center',
            template: '#template', width: 120
        },
        { field: 'ProductID', headerText: 'Product ID', textAlign:
'Right', width: 80 },
        { field: 'ProductName', headerText: 'Name', width: 160 },
        { field: 'SupplierID', headerText: 'SupplierID', width: 80 },
        { field: 'UnitsInStock', headerText: 'Stock', width: 80,
textAlign: 'Right' }
    ],
    height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```

<body>

  <div id="container">
    <script id="template" type="text/x-template">
      <div class="template_checkbox">
        ${if(Discontinued)}
        <input type="checkbox" checked> ${else}
        <input type="checkbox"> ${/if}
      </div>
    </script>
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to get the row object by clicking on the template element

You can get the row object without selecting the row and achieve it using the column template feature and the `getRowObjectFromUID` method of the Grid.

In the following sample, the button element is rendered in the Employee Data column. By clicking the button, you can get the row object using the `getRowObjectFromUID` method of the Grid and display it in the console.

INDEX.TS

```

import { Grid, RecordClickEventArgs } from '@syncfusion/ej2-grids';
import { employeeData } from './datasource.ts';
import { closest } from '@syncfusion/ej2-base';
let grid: Grid = new Grid({
  dataSource: employeeData,
  columns: [
    {
      headerText: 'Employee Data', textAlign: 'Right',
      template: '#template', width: 150, isPrimaryKey: true
    },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 130 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'Title', headerText: 'Title', width: 170 }
  ],
  height: 315,
  recordClick: (args: RecordClickEventArgs) => {
    if (args.target.classList.contains('empData')) {
      var rowObj = grid.getRowObjectFromUID(closest(args.target, '.e-
row').getAttribute('data-uid')
    );
      console.log(rowObj);
    }
  }
});

```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <script id="template" type="text/x-template">
      <button class="empData">Employee Data</button>
    </script>
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Complex data binding in EJ2 JavaScript Grid control

You can achieve complex data binding in the grid by using the dot(.) operator in the [column.field](#).

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { complexData } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: complexData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 100 },
        { field: 'Name.FirstName', headerText: 'First Name', width: 120 },
        { field: 'Name.LastName', headerText: 'Last Name', width: 100 },
        { field: 'Title', headerText: 'Title', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

For OData and ODataV4 adaptors, you need to add [expand](#) query to the [query](#) property (of Grid), to eager load the complex data.

`ts

```

import { DataManager, ODataAdaptor, Query } from '@syncfusion/ej2-data';

let data: DataManager = new DataManager({
    adaptor: new ODataAdaptor(),
    crossDomain: true,
    url: 'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders'
});

let query: Query = new Query().expand('Employee');

let grid: Grid = new Grid({
    dataSource: data,
    query: query,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'Employee.City', headerText: 'Employee City', width: 150 }
    ]
});

```

```

],
height: 315
});
grid.appendTo('#Grid');
`

```

Foreign key column in EJ2 JavaScript Grid control

The Foreign key column in the Syncfusion Grid control allows you to display related data from a foreign key data source in a column within the grid. This feature is particularly useful when you have a column in the grid that represents a foreign key relationship with another data source.

To enable and integrate the foreign key column in the EJ2 JavaScript Grid control, follow these steps:

1. Inject the ForeignKeyService in the Grid

```

`typescript
ej.grids.Grid.Inject(ej.grids.ForeignKey);
`

```

2. Define the foreign key column in the grid using the following properties: * [dataSource](#): Specifies the foreign data source that contains the related data. * [foreignKeyField](#): Maps the column name in the grid to the field in the foreign data source that represents the foreign key relationship. * [foreignKeyValue](#): Specifies the field from the foreign data source that should be displayed in the grid as the related data.

```

`typescript
{field: 'EmployeeID', headerText: 'Employee ID', width: 150, foreignKeyValue: 'FirstName',
foreignKeyField: 'EmployeeID', dataSource: employeeData}
`

```

The `foreignKeyField` property should match the name of the field in the foreign data source that represents the foreign key relationship, and the `foreignKeyValue` property should specify the field from the foreign data source that should be displayed in the grid as the related data.

Binding local data

The Syncfusion Grid control provides a convenient way to bind local data to a foreign key column. This allows you to display related data from a local data source within the grid. Here's an example of how to bind local data to a Foreign Key column in Syncfusion Grid:

In this example, **data** is the local data source for the Grid, and **employeeData** is the local data source for the foreign key column. The `field` property of the column is set to **EmployeeID** which represents the foreign key value in the **data**. The `foreignKeyValue` property is set to **FirstName** which represents the field name in the **employeeData** that you want to display in the foreign key column.

INDEX.TS

```

import { Grid, ForeignKey } from '@syncfusion/ej2-grids';
import { data, employeeData } from './datasource.ts';
Grid.Inject(ForeignKey);
let grid: Grid = new Grid(
{
    dataSource: data,

```



```

        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
            {field: 'EmployeeID', headerText: 'Employee Name', width: 120,
foreignKeyValue: 'FirstName', dataSource: employeeData},
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 80},
            { field: 'ShipCity', headerText: 'Ship City', width: 130 },
        ],
        height: 315
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
    <script>
        var ele = document.getElementById('container');

```

```

        if (ele) {
            ele.style.visibility = "visible";
        }
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Binding remote data

The Foreign key column in Syncfusion Grid allows you to bind remote data for a foreign key column. You can assign the service data as an instance of `DataManager` to the `dataSource` property, and provide the endpoint `URL` as the data source URL.

This example demonstrates how to use the foreign key column with remote data binding using the [ODataV4Adaptor](#) in the grid:

INDEX.TS

```

import { Grid, ForeignKey } from '@syncfusion/ej2-grids';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/',
    adaptor: new ODataV4Adaptor
});
let employeeData: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Employees/',
    adaptor: new ODataV4Adaptor
});
Grid.Inject(ForeignKey);
let grid: Grid = new Grid(
    {
        dataSource: data,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
            {field: 'EmployeeID', headerText: 'Employee Name', width: 120,
foreignKeyValue: 'FirstName', dataSource: employeeData},
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 80},
            { field: 'ShipCity', headerText: 'Ship City', width: 130 },
        ],
        height: 315
    });
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container">
<div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if (ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

* For remote data, the sorting and grouping is done based on [column.foreignKeyField](#) instead of [column.foreignKeyValue](#).

* If [column.foreignKeyField](#) is not defined, then the column uses [column.field](#).

Use edit template in foreignkey column

The Syncfusion Grid provides support for using an edit template in a foreign key column. By default, a dropdown control is used for editing foreign key column. However, you can render a different control for editing by using the [column.edit](#) property. Here's an example that demonstrates how to use an edit template in a foreign key column:

In this example, an [AutoComplete](#) control is rendered as the edit template for the “EmployeeID” foreign key column. The [dataSource](#) property of the [AutoComplete](#) control is set to the employees data, and the [fields](#) property is configured to display the “FirstName” field as the value.

INDEX.TS

```

import { createElement } from '@syncfusion/ej2-base';
import { Grid, ForeignKey, Edit, Toolbar, ColumnModel } from
 '@syncfusion/ej2-grids';
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
import { DataManager, Query } from '@syncfusion/ej2-data';
import { data, employeeData } from './datasource.ts';
Grid.Inject(ForeignKey, Edit, Toolbar);
let autoComplete: AutoComplete;
let grid: Grid = new Grid(
    {
        dataSource: data,
        editSettings: { allowEditing: true },
        toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
        columns: [
            { field: 'OrderID', headerText: 'Order ID', isPrimaryKey:
true, textAlign: 'Right', width: 100 },
            // Foreign column
            {
                field: 'EmployeeID', headerText: 'First Name', width:
150, foreignKeyValue: 'FirstName', dataSource: employeeData,
                edit: {
                    create: () => { // to create input element
                        return createElement('input');
                    },
                    read: () => { // return edited value to update
data source
                        let value: Object[] = new
DataManager(employeeData).executeLocal(new Query().where('FirstName',
'equal', autoComplete.value));
                        return value.length &&
value[0]['EmployeeID']; // to convert foreign key value to local value.
                    },
                    destroy: () => { // to destroy the custom
component.
                        autoComplete.destroy();
                    },
                    write: (args: { rowData: Object, column:
ColumnModel, foreignKeyData: Object, element: HTMLElement }) => { // to show
the value for custom component
                        autoComplete = new AutoComplete({
                            dataSource: employeeData,
                            fields: { value:
args.column.foreignKeyValue },
                            value:
args.foreignKeyData[0][args.column.foreignKeyValue]
                        });
                        autoComplete.appendTo(args.element);
                    }
                }
            },
            { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 80 },
            { field: 'ShipCity', headerText: 'Ship City', width: 130 }
        ],
        height: 270
    }

```

```
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Customize filter UI in foreignkey column

The Syncfusion Grid allows you to customize the filtering user interface (UI) for foreign key columns by using the [column.filter](#) property. By default, a dropdown control is used for filtering foreign key columns. However, you can create your own custom filtering UI by specifying a template function for the [column.filter](#) property. Here's an example that demonstrates how to create a custom filtering UI in a foreign key column:

In this example, a [DropDownList](#) control is rendered as the filter UI for the “EmployeeID” foreign key column. The [dataSource](#) property of the DropDownList control is set to the employees data, and the [fields](#) property is configured to display the **FirstName** field as the [text](#) and **EmployeeID** field as the [value](#). The [value](#) property is set to the current filter value of the column.

INDEX.TS

```
import { createElement } from '@syncfusion/ej2-base';
import { DataManager } from '@syncfusion/ej2-data';
import { Grid, ForeignKey, Filter, ColumnModel } from '@syncfusion/ej2-grids';
import { data, fEmployeeData } from './datasource.ts';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
Grid.Inject(ForeignKey, Filter);
let dropInstance;
let grid: Grid = new Grid(
    {
        dataSource: data,
        allowFiltering: true,
        filterSettings: { type: 'Menu' },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
            // Foreign column
            {
                field: 'EmployeeID', headerText: 'Employee Name', width:
150, foreignKeyValue: 'FirstName', dataSource: fEmployeeData,
                filter: {
                    ui: {
                        create: (args: { target: Element, column: Object
})) => {
                            let flValInput: HTMLElement =
createElement('input', { className: 'flm-input' });
                            args.target.appendChild(flValInput);
                            dropInstance = new DropDownList({
                                dataSource: new
DataManager(fEmployeeData ),
                                fields: { text: 'FirstName', value:
'EmployeeID' },
                                placeholder: 'Select a value',
                                popupHeight: '200px'
                            });
                            dropInstance.appendTo(flValInput);
                        },
                        write: (args: {
                            column: Object, target: Element, parent:
Element,
                            filteredValue: number | string
                        }) => {
```

```

dropInstance.text = args.filteredValue ||
'';
    },
    read: (args: { target: Element, column:
ColumnModel, operator: string, fltrObj: Filter }) => {
args.fltrObj.filterByColumn(args.column.field, args.operator,
dropInstance.text);
    }
    }
    },
    { field: 'Freight', headerText: 'Freight', width: 100,
textAlign: 'Right'},
    { field: 'ShipCity', headerText: 'Ship City', width: 180 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container">
<div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if (ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Use filter bar template in foreignkey column

You can use the filter bar template in a foreign key column in Grid by defining the [column.filterBarTemplate](#) property. This allows you to customize the filter bar for the foreign key column with a custom control or HTML template. Here's an example that demonstrates how to use a filter bar template in a foreign key column:

In this example, the “**EmployeeID**” column is a foreign key column, and the **filter** function is used as the filter bar template for this column. The **filter** function can be defined in your control code and should return the desired control or HTML template for the filter bar. The column header shows the custom filter bar template and you can select filter value by using the **DropDown** options.

INDEX.TS

```

import { Grid, ForeignKey, Filter , Column} from '@syncfusion/ej2-grids';
import { data, fEmployeeData } from './datasource.ts';
import { createElement } from '@syncfusion/ej2-base';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { DataManager } from '@syncfusion/ej2-data';
Grid.Inject(ForeignKey, Filter);
let grid: Grid = new Grid(
{
  dataSource: data,
  allowFiltering: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    // Foreign column
    {
      field: 'EmployeeID', headerText: 'Employee Name', width:
120, foreignKeyValue: 'FirstName', dataSource: fEmployeeData,
      filterBarTemplate: {
        create: () =>
        {

```



```

        return createElement('input', { className: 'flm-
input' });};
    },
    write: (args: { element: Element, column: Column }) => {
        fEmployeeData.splice(0, 0, {'FirstName': 'All'}); //
for clear filtering
        let dropInstance: DropDownList = new DropDownList({
            dataSource: new DataManager(fEmployeeData),
            fields: { text: 'FirstName' },
            placeholder: 'Select a value',
            popupHeight: '200px',
            index: 0,
            change: (args: { value: string }) => {
                if (args.value !== 'All') {
                    grid.filterByColumn('EmployeeID',
'equal', args.value);
                }
                else {
grid.removeFilteredColsByField('EmployeeID');
                }
            }
        });
        dropInstance.appendTo(args.element);
    }
},
{ field: 'ShipCity', headerText: 'Ship City', width: 180 },
],
height: 260
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Perform aggregation in foreignkey column

By default, aggregations are not supported in a foreign key column in the Syncfusion Grid. However, you can achieve aggregation for a foreign key column by using [customAggregate](#).

To perform aggregation in a foreign key column, follow these steps:

1. Define a foreign key column in the Grid.
2. Implement a custom aggregate function to calculate the aggregation for the foreign key column.
3. Set the [customAggregate](#) property of the column to the custom aggregate function.

Here's an example that demonstrates how to perform aggregation in a foreign key column:

In the provided example, the `customAggregateFn` function is used to filter the data based on the **FirstName** field of the foreign key column, using the `getForeignData` internal function. The function then counts the occurrences of **Margaret**. The result is displayed in the grid's footer template using the `footerTemplate`.

INDEX.TS

```

import { Grid, ForeignKey, Aggregate, getForeignData, CustomSummaryType,
AggregateColumnModel } from '@syncfusion/ej2-grids';
import { data, employeeData } from './datasource.ts';
import { getValue } from '@syncfusion/ej2-base';
Grid.Inject(ForeignKey, Aggregate);
// Custom Aggregate function for foreign column
let customAggregateFn: CustomSummaryType = (data1: { result: { OrderID:
number, EmployeeID: number, Freight: number, ShipName: string }[] }, column:
AggregateColumnModel) => {

```

```

        return data1.result.filter((count: { OrderID: number, EmployeeID:
number, Freight: number, ShipName: string }) => {
            return getValue('FirstName',
getForeignData(grid.getColumnByField(column.columnName), count)[0]) ===
'Margaret';
        }).length;
    };
let grid: Grid = new Grid(
    {
        dataSource: data,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
            // Foreign column
            {field: 'EmployeeID', headerText: 'Employee Name', width: 150,
foreignKeyValue: 'FirstName', dataSource: employeeData,},
            { field: 'Freight', headerText: 'Freight', width: 100,
textAlign: 'Right'},
            { field: 'ShipName', headerText: 'Ship Name', width: 180 }
        ],
        height: 280,
        aggregates: [
            {
                columns: [
                    {
                        type: 'Custom',
                        customAggregate: customAggregateFn,
                        field: 'EmployeeID',
                        footerTemplate: 'Count of Margaret: ${Custom}'
                    }
                ]
            }
        ]
    }
);
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Enable multiple foreign key columns

The Syncfusion Grid control supports the feature of enabling multiple foreign key columns with editing options. This allows users to display columns from foreign data sources in the Grid control.

In the following example, **Customer Name** and **Ship City** are foreign key columns that display the **ContactName** and **City** columns from foreign data.

INDEX.TS

```

import { Grid, Page, Edit, Toolbar, ForeignKey } from '@syncfusion/ej2-
grids';
import { orderDetails, customerData, employeeData } from './datasource.ts';
Grid.Inject(Page, Edit, Toolbar, ForeignKey);
let grid: Grid = new Grid(
  {
    dataSource: orderDetails,
    allowPaging: true,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true,
allowDeleting: true },
    columns: [
      { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
isPrimaryKey: true, width: 100 },

```

```

        { field: 'CustomerID', foreignKeyField: 'CustomerID',
foreignKeyValue: 'ContactName', dataSource: customerData, width: 170,
headerText: 'Customer Name', validationRules: { required: true } },
        { field: 'Freight', headerText: 'Freight', editType:
'numericedit', width: 130, textAlign: 'Right', format: 'C2' },
        { field: 'EmployeeID', foreignKeyField: 'EmployeeID',
foreignKeyValue: 'City', dataSource: employeeData, width: 150, headerText:
'Ship City', validationRules: { required: true } },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
</script>

```

```

        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Auto fit columns in EJ2 JavaScript Grid control

The [autoFitColumns](#) method resizes the column to fit the widest cell's content without wrapping. You can autofit a specific column at initial rendering by invoking the [autoFitColumns](#) method in [dataBound](#) event.

To use the [autoFitColumns](#) method, inject the **Resize** module in the grid.

INDEX.TS

```

import { Grid, Resize } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Resize);
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 140 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 130 },
        { field: 'ShipName', headerText: 'Ship Name', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 120 },
        { field: 'ShipAddress', headerText: 'Ship Address', width: 150 }
    ],
    dataBound: () => grid.autoFitColumns(['ShipName', 'ShipAddress']),
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can autofit all the columns by invoking the [autoFitColumns](#) method without column names.

Column reorder in EJ2 JavaScript Grid control

Reordering can be done by drag and drop of a particular column header from one index to another index within the grid. To enable reordering, set the [allowReordering](#) to true.

To use reordering, inject the [Reorder](#) module in the grid.

INDEX.TS

```

import { Grid, Reorder } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Reorder);
let grid: Grid = new Grid({
    dataSource: data,
    allowReordering: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },

```

```

        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script id="rowtemplate" type="text/x-template">
        <tr>
            <td class="photo">
                
            </td>

```



```

        <td class="details">
            <table class="CardTable" cellpadding="3" cellspacing="2">
                <colgroup>
                    <col width="50%">
                    <col width="50%">
                </colgroup>
                <tbody>
                    <tr>
                        <td class="CardHeader">First Name </td>
                        <td>${FirstName} </td>
                    </tr>
                    <tr>
                        <td class="CardHeader">Last Name</td>
                        <td>${LastName} </td>
                    </tr>
                    <tr>
                        <td class="CardHeader">Title
                        </td>
                        <td>${Title}
                        </td>
                    </tr>
                    <tr>
                        <td class="CardHeader">Country
                        </td>
                        <td>${Country}
                        </td>
                    </tr>
                </tbody>
            </table>
        </td>
    </tr>
</script>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can disable reordering a particular column by setting the [columns.allowReordering](#) to **false**.

Reorder single column

Grid have option to reorder Columns either by Interaction or by using the [reorderColumns](#) method. In the below sample, **ShipCity** column is reordered to last column position by using the method.

INDEX.TS

```

import { Grid, Reorder } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';

```

```

Grid.Inject(Reorder);
let grid: Grid = new Grid({
    dataSource: data,
    allowReordering: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipRegion', headerText: 'Ship Region', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 280
});
grid.appendTo('#Grid');
let reorderSingleColsBtn: Button = new Button();
reorderSingleColsBtn.appendTo('#reorderSingleCol');
document.getElementById('reorderSingleCol').addEventListener('click', () =>
{
    grid.reorderColumns('ShipCity', 'ShipName');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
#reorderMultipleCols {
    text-transform: none;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<script id="rowtemplate" type="text/x-template">
<tr>
<td class="photo">

</td>
<td class="details">
<table class="CardTable" cellpadding="3" cellspacing="2">
<colgroup>
<col width="50%">
<col width="50%">
</colgroup>
<tbody>
<tr>
<td class="CardHeader">First Name </td>
<td>{FirstName} </td>
</tr>
<tr>
<td class="CardHeader">Last Name</td>
<td>{LastName} </td>
</tr>
<tr>
<td class="CardHeader">Title
</td>
<td>{Title}
</td>
</tr>
<tr>
<td class="CardHeader">Country
</td>
<td>{Country}
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</script>

<div id="container">
<button id="reorderSingleCol">Reorder Ship City to Last</button>
<div id="Grid"></div>
</div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Reorder multiple columns

User can reorder a single column at a time by Interaction. Sometimes we need to have reorder multiple columns at the same time, It can be achieved through programmatically by using [reorderColumns](#) method.

In the below sample, **Ship City** and **Ship Region** column is reordered to last column position.

INDEX.TS

```

import { Grid, Reorder } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
Grid.Inject(Reorder);
let grid: Grid = new Grid({
    dataSource: data,
    allowReordering: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipRegion', headerText: 'Ship Region', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 280
});
grid.appendTo('#Grid');
let reorderMultipleColsBtn: Button = new Button();
reorderMultipleColsBtn.appendTo('#reorderMultipleCols');
document.getElementById('reorderMultipleCols').addEventListener('click', ()
=> {
    grid.reorderColumns(['ShipCity', 'ShipRegion'], 'ShipName');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
#reorderMultipleCols {
    text-transform: none;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<script id="rowtemplate" type="text/x-template">
<tr>
<td class="photo">

</td>
<td class="details">
<table class="CardTable" cellpadding="3" cellspacing="2">
<colgroup>
<col width="50%">
<col width="50%">
</colgroup>
<tbody>
<tr>
<td class="CardHeader">First Name </td>
<td>{FirstName} </td>
</tr>
<tr>
<td class="CardHeader">Last Name</td>
<td>{LastName} </td>
</tr>
<tr>
<td class="CardHeader">Title

```

```

        </td>
        <td>${Title}
        </td>
    </tr>
    <tr>
        <td class="CardHeader">Country
        </td>
        <td>${Country}
        </td>
    </tr>
</tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <button id="reorderMultipleCols">Reorder Ship City and Ship Region
    to Last</button>
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Reorder events

During the reorder action, the grid component triggers the below three events.

1. The [columnDragStart](#) event triggers when column header element drag (move) starts.
2. The [columnDrag](#) event triggers when column header element is dragged (moved) continuously.
3. The [columnDrop](#) event triggers when a column header element is dropped on the target column.

INDEX.TS

```

import { Grid, Reorder } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Reorder);
let grid: Grid = new Grid({
    dataSource: data,
    allowReordering: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    columnDragStart: () => {

```

```

        alert('columnDragStart event is Triggered');
    },
    columnDrag: () => {
        alert('columnDrag event is Triggered');
    },
    columnDrop: () => {
        alert('columnDrop event is Triggered');
    }
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">

```

```

<tr>
  <td class="photo">
    
  </td>
  <td class="details">
    <table class="CardTable" cellpadding="3" cellspacing="2">
      <colgroup>
        <col width="50%">
        <col width="50%">
      </colgroup>
      <tbody>
        <tr>
          <td class="CardHeader">First Name </td>
          <td>${FirstName} </td>
        </tr>
        <tr>
          <td class="CardHeader">Last Name</td>
          <td>${LastName} </td>
        </tr>
        <tr>
          <td class="CardHeader">Title
          </td>
          <td>${Title}
          </td>
        </tr>
        <tr>
          <td class="CardHeader">Country
          </td>
          <td>${Country}
          </td>
        </tr>
      </tbody>
    </table>
  </td>
</tr>
</script>

<div id="container">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column resizing in EJ2 JavaScript Grid control

Column width can be resized by clicking and dragging the right edge of the column header. While dragging, the width of the respective column will be resized immediately. Each column can be auto resized by double-clicking the right edge of the column header to fit the width of that column based on the widest cell content. To enable column resize, set the [allowResizing](#) property to true.

To use the column resize, inject **Resize** module in the grid.

INDEX.TS

```
import { Grid, Resize } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Resize);
let grid: Grid = new Grid({
    dataSource: data,
    allowResizing: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width:150 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'Freight', headerText: 'Freight', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 },
        { field: 'ShipAddress', headerText: 'Ship Address', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipPostalCode', headerText: 'Ship Postal code', width:
150 }
    ],
    height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* You can disable resizing for a particular column by setting the [columns.allowResizing](#) to false.

* In RTL mode, you can click and drag the left edge of the header cell to resize the column.

Column resizing externally

To resize a column, set width to that particular column and then refresh the grid header by using the [refreshHeader\(\)](#) method. Please refer the below code

```

`ts
var grid = document.getElementById('Grid').ej2_instances[0]; //Grid Instance
var columns = grid.columns;
columns[0].width = 150;
grid.refreshHeader();
`

```

Min and max width

Column resize can be restricted between minimum and maximum width by defining the [columns->minWidth](#) and [columns->maxWidth](#).

In the following sample, minimum and maximum width are defined for **OrderID**, **Ship Name**, and **Ship Country** columns.

INDEX.TS

```

import { Grid, Resize } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Resize);
let grid: Grid = new Grid({
  dataSource: data,
  allowResizing: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
minWidth: 100, width: 150, maxWidth: 300 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'Freight', headerText: 'Freight', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', minWidth: 120, width:
150, maxWidth: 250 },
    { field: 'ShipAddress', headerText: 'Ship Address', width: 150 },
    { field: 'ShipCountry', headerText: 'Ship Country', minWidth: 150,
width: 200, maxWidth: 350 },
    { field: 'ShipPostalCode', headerText: 'Ship Postal code', width:
150 }
  ],
  height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>

```

```

        <tr>
            <td class="CardHeader">First Name </td>
            <td>${FirstName} </td>
        </tr>
        <tr>
            <td class="CardHeader">Last Name</td>
            <td>${LastName} </td>
        </tr>
        <tr>
            <td class="CardHeader">Title
            </td>
            <td>${Title}
            </td>
        </tr>
        <tr>
            <td class="CardHeader">Country
            </td>
            <td>${Country}
            </td>
        </tr>
    </tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The `maxWidth` and `minWidth` properties will be considered only when the user resizes the column. When resizing the window, these properties will not be considered. This is because columns cannot be re-rendered when resizing the window.

Resize stacked column

Stacked columns can be resized by clicking and dragging the right edge of the stacked column header. While dragging, the width of the respective child columns will be resized at the same time. You can disable resize for any particular stacked column by setting `allowResizing` as **false** to its columns.

In this example, we have disabled resize for **Ship City** column.

INDEX.TS

```

import { Grid, Resize } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Resize);
let grid: Grid = new Grid({
    dataSource: data,

```

```

        allowResizing: true,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, minWidth: 10 },
            {
                headerText: 'Order Details', columns: [
                    { field: 'OrderDate', headerText: 'Order Date',
textAlign: 'Right', width: 125, minWidth: 10, format: 'yMd' },
                    { field: 'Freight', headerText: 'Freight($)', textAlign:
'Right', width: 100, format: 'C2', minWidth: 10 },
                ]
            },
            {
                headerText: 'Ship Details', columns: [
                    { field: 'ShipCity', headerText: 'Ship City', width:
100, minWidth: 10, allowResizing: false },
                    { field: 'ShipCountry', headerText: 'Ship Country',
width: 120, minWidth: 10 },
                ]
            }
        ]
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>${FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>${LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>${Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>${Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

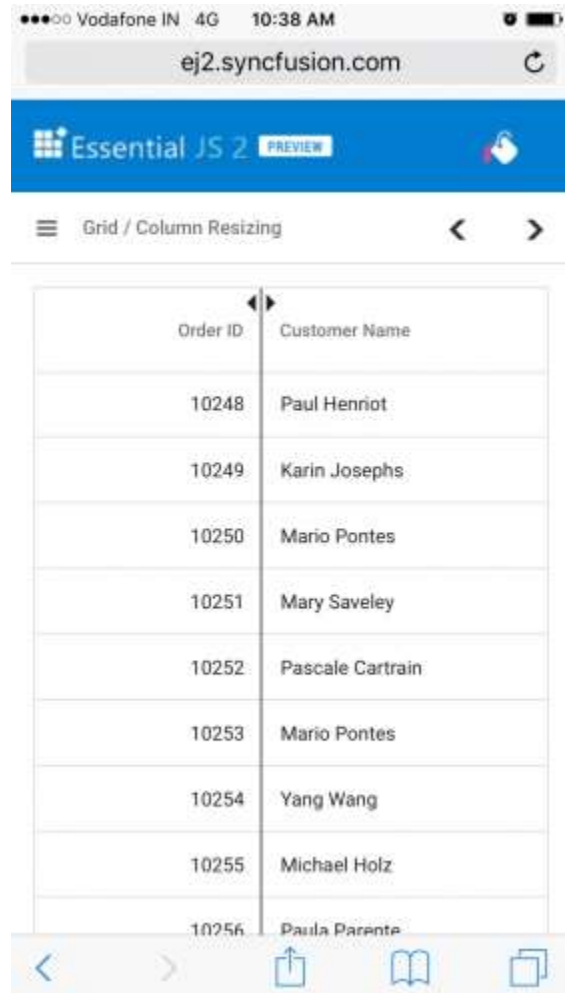
```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Touch interaction

When the right edge of the header cell is tapped, a floating handler will be visible over the right border of the column. To resize the column, tap and drag the floating handler as needed. You can autoFit a column by using the Column menu of the grid.

The following screenshot represents the column resizing in touch device.



Resizing events

During the resizing action, the grid component triggers the below three events.

1. The [resizeStart](#) event triggers when column resize starts.
2. The [resizing](#) event triggers when column header element is dragged (moved) continuously..
3. The [resizeStop](#) event triggers when column resize ends.

INDEX.TS

```
import { Grid, Resize } from '@syncfusion/ej2-grids';
```



```

import { data } from './datasource.ts';
Grid.Inject(Resize);
let grid: Grid = new Grid({
    dataSource: data,
    allowResizing: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width:150 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'Freight', headerText: 'Freight', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 },
        { field: 'ShipAddress', headerText: 'Ship Address', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipPostalCode', headerText: 'Ship Postal code', width:
150 }
    ],
    height: 315,
    resizeStart: () => {
        alert('resizeStart event is Triggered');
    },
    resizing: () => {
        alert('Resizing event is Triggered');
    },
    resizeStop: () => {
        alert('resizeStop event is Triggered');
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column chooser in EJ2 JavaScript Grid control

The column chooser feature in the Syncfusion EJ2 JavaScript Grid control allows you to dynamically show or hide columns. This feature can be enabled by defining the [showColumnChooser](#) property as **true**.

To use the column chooser, inject the **ColumnChooser** module in the grid.

INDEX.TS

```

import { Grid, Toolbar, ColumnChooser } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ColumnChooser);
let grid: Grid = new Grid({
    dataSource: data,
    showColumnChooser: true,
    toolbar: ['ColumnChooser'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right' },
        { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
        { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
textAlign: 'Right' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
150 }
    ],
    height: 235
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The column chooser dialog displays the header text of each column by default. If the header text is not defined for a column, the corresponding column field name is displayed instead.

Hide column in column chooser dialog

You can hide the column names in column chooser by defining the [columns->showInColumnChooser](#) as **false**. This feature is useful when working with a large number of columns or when you want to limit the number of columns that are available for selection in the column chooser dialog.

In this example, the `columns->showInColumnChooser` property is set to false for the **Order ID** column. As a result, the **Order ID** column will not be displayed in the column chooser dialog.

INDEX.TS

```

import { Grid, Toolbar, ColumnChooser } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject( Toolbar, ColumnChooser );
let grid: Grid = new Grid({
  dataSource: data,
  showColumnChooser: true,

```

```

        toolbar: ['ColumnChooser'],
        columns: [
            { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right', showInColumnChooser: false },
            { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
            { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
textAlign: 'Right' },
            { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
150 }
        ],
        height: 272
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

The `columns->showInColumnChooser` property is applied to each column element individually. By setting it to false, you can hide specific columns from the column chooser dialog.

Open column chooser by external button

The Syncfusion EJ2 JavaScript Grid provides the flexibility to open the column chooser dialog on a web page using an external button. By default, the column chooser button is displayed in the right corner of the grid control, and clicking the button opens the column chooser dialog below it. However, you can programmatically open the column chooser dialog at specific **X** and **Y** axis positions by using the [openColumnChooser](#) method.

Here's an example of how to open the column chooser in the Grid using an external button:

INDEX.TS

```
import { Grid, ColumnChooser } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
Grid.Inject( ColumnChooser);
let grid: Grid = new Grid({
    dataSource: data,
    showColumnChooser: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right', showInColumnChooser: false },
        { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
        { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
textAlign: 'Right' },
        { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
150 }
    ],
    height: 235
});
grid.appendTo('#Grid');
let showButton: Button = new Button({ cssClass: 'e-primary' }, '#show');
(document.getElementById('show') as HTMLElement).onclick = () => {
    grid.columnChooserModule.openColumnChooser(100, 40); // give X and Y
axis
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en">
<head>
```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <button id="show">Open Column Chooser</button>
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize column chooser dialog size

The column chooser dialog in Syncfusion EJ2 JavaScript Grid comes with default size, but you can modify its height and width as per your specific needs using CSS styles.

To customize the column chooser dialog size, you can use the following CSS styles:

```

`css
.e-grid .e-dialog.e-ccdlg {

```

```

height: 500px;
width: 200px;
}
.e-grid .e-ccdlg .e-cc-contentdiv {
height: 200px;
width: 230px;
}
`

```

INDEX.TS

```

import { Grid, Toolbar, ColumnChooser } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject( Toolbar, ColumnChooser);
let grid: Grid = new Grid({
    dataSource: data,
    showColumnChooser: true,
    toolbar: ['ColumnChooser'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right', showInColumnChooser: false },
        { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
        { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
textAlign: 'Right' },
        { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
150 }
    ],
    height: 235
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<style>
    .e-grid .e-dialog.e-ccdlg {
        height: 500px;
        width: 200px;
    }
    .e-grid .e-ccdlg .e-cc-contentdiv {
        height: 200px;
        width: 230px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change default search operator of the column chooser

The column chooser dialog in the Syncfusion EJ2 JavaScript Grid provides a search box that allows you to search for column names. By default, the search functionality uses the "startswith" operator to match columns and display the results in the column chooser dialog. However, there might be cases where you need to change the default search operator to achieve more precise data matching.

To change the default search operator of the column chooser in Syncfusion Grid, you need to use the [operator](#) property of the `columnChooserSettings`.

Here's an example of how to change the default search operator of the column chooser to **contains** in the EJ2 JavaScript Grid:

INDEX.TS

```
import { Grid, Toolbar, ColumnChooser } from '@syncfusion/ej2-grids';
```

```

import { data } from './datasource.ts';
Grid.Inject( Toolbar, ColumnChooser);
let grid: Grid = new Grid({
    dataSource: data,
    showColumnChooser: true,
    toolbar: ['ColumnChooser'],
    columnChooserSettings: { operator: 'contains' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right' },
        { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd', textAlign: 'Right' },
        { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
textAlign: 'Right' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 130 },
        { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
130 }
    ],
    height: 235
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```

```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Diacritics searching in column chooser

By default, the grid ignores diacritic characters when performing a search in the column chooser. However, in some cases, you may want to include diacritic characters in the search. To enable this behavior, you can set the [columnChooserSettings->ignoreAccent](#) property to **true**.

Here is an example that demonstrates the usage of the `ignoreAccent` property to include diacritic characters for searching in the column chooser:

INDEX.TS

```
import { Grid, Toolbar, ColumnChooser } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject( Toolbar, ColumnChooser);
let grid: Grid = new Grid({
  dataSource: data,
  showColumnChooser: true,
  toolbar: ['ColumnChooser'],
  columnChooserSettings : { ignoreAccent: true },
  columns: [
    { field: 'OrderID^', headerText: 'Order ID^', width: 120, textAlign:
'Right' },
    { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd', textAlign: 'Right' },
    { field: 'F^reight', headerText: 'F^reight', width: 120, format:
'C2', textAlign: 'Right' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 130 },
    { field: 'ShipCity', headerText: 'Ship City', visible: false, width:
130 }
  ],
  height: 272
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column menu in EJ2 JavaScript Grid control

The column menu has options to integrate features like sorting, grouping, filtering, column chooser, and autofit. It will show a menu with the integrated feature when users click on multiple icon of the column. To enable column menu, you need to define the [showColumnMenu](#) property as true.

To use the column menu, inject the **ColumnMenu** module in the grid.

The default items are displayed in following table.

Item	Description
-----	-----

- | **SortAscending** | Sort the current column in ascending order. |
- | **SortDescending** | Sort the current column in descending order. |
- | **Group** | Group the current column. |
- | **Ungroup** | Ungroup the current column. |
- | **AutoFit** | Auto fit the current column. |
- | **AutoFitAll** | Auto fit all columns. |
- | **ColumnChooser** | Choose the column visibility. |
- | **Filter** | Show the filter option as given in [filterSettings.type](#) |

INDEX.TS

```
import { Grid, Resize, Sort, Group, Filter, ColumnMenu, Page } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Resize, Sort, Group, Filter, ColumnMenu, Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowGrouping: true,
  allowSorting: true,
  allowFiltering: true,
  filterSettings: { type: 'CheckBox' },
  allowPaging: true,
  groupSettings: { showGroupedColumn: true },
  showColumnMenu: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 200, textAlign:
    'Right',
      showInColumnChooser: false },
    { field: 'Freight', width: 150, format: 'C2', textAlign: 'Right',
    editType: 'numericedit' },
    { field: 'ShipName', headerText: 'Ship Name', width: 300 },
    { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
    width: 200 },
    { field: 'ShipCity', headerText: 'Ship City', width: 200 }
  ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

```

```

                <td class="CardHeader">Country
                </td>
                <td>${Country}
                </td>
            </tr>
        </tbody>
    </table>
</td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* You can disable column menu for a particular column by defining the [columns.showColumnMenu](#) as false.

* You can customize the default items by defining the [columnMenuItems](#) with required items.

Column menu events

During the resizing action, the grid component triggers the below two events.

1. The [columnMenuOpen](#) event triggers before the column menu opens.
2. The [columnMenuClick](#) event triggers when the user clicks the column menu of the grid.

INDEX.TS

```

import { Grid, Resize, Sort, Group, Filter, ColumnMenu, Page } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Resize, Sort, Group, Filter, ColumnMenu, Page);
let grid: Grid = new Grid({
    dataSource: data,
    allowGrouping: true,
    allowSorting: true,
    allowFiltering: true,
    filterSettings: { type: 'CheckBox' },
    allowPaging: true,
    groupSettings: { showGroupedColumn: true },
    showColumnMenu: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 200, textAlign:
 'Right',
            showInColumnChooser: false },
        { field: 'Freight', width: 150, format: 'C2', textAlign: 'Right',
 editType: 'numericedit' },
    ]

```

```

        { field: 'ShipName', headerText: 'Ship Name', width: 300 },
        { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
width: 200 },
        { field: 'ShipCity', headerText: 'Ship City', width: 200 }
    ],
    columnMenuOpen: () => {
        alert('columnMenuOpen event is Triggered');
    },
    columnMenuClick: () => {
        alert('columnMenuClick event is Triggered');
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```



```

<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom column menu item

Custom column menu items can be added by defining the [columnMenuItems](#) as collection of the [columnMenuItemModel](#). Actions for this customized items can be defined in the [columnMenuClick](#) event.

INDEX.TS

```
import { Grid, ColumnMenu, Sort, Page } from '@syncfusion/ej2-grids';
import { MenuEventArgs } from '@syncfusion/ej2-navigations';
import { data } from './datasource.ts';
Grid.Inject(ColumnMenu, Page, Sort);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowSorting: true,
    showColumnMenu: true,
    columnMenuItems: [{text: 'Clear Sorting', id: 'gridclearsorting'}],
    columnMenuClick: function(args: MenuEventArgs) {
        if(args.item.id === 'gridclearsorting') {
            grid.clearSorting();
        }
    },
    sortSettings: {
        columns: [{direction: "Ascending", field: "OrderID"}]
    },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 200, textAlign: 'Right', showInColumnChooser: false },
        { field: 'Freight', width: 150, format: 'C2', textAlign: 'Right', editType: 'numericedit' },
        { field: 'ShipName', headerText: 'Ship Name', width: 300 },
        { field: 'ShipCountry', headerText: 'Ship Country', visible: false, width: 200 },
        { field: 'ShipCity', headerText: 'Ship City', width: 200 }
    ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>${FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>${LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>${Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>${Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

```

```

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize menu items for particular columns

Sometimes, you have a scenario that to hide an item from column menu for particular columns. In that case, you need to define the [columnMenuOpenEventArgs.hide](#) as true in the [columnMenuOpen](#) event.

The following sample, **Filter** item was hidden in column menu when opens for the **OrderID** column.

INDEX.TS

```

import { Grid, ColumnMenu, Sort, Page, Resize, Group, Filter,
ColumnMenuOpenEventArgs, ColumnMenuItemModel } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(ColumnMenu, Page, Group, Sort, Resize, Filter);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    showColumnMenu: true,
    filterSettings: { type: 'Menu' },
    allowFiltering: true,
    allowGrouping: true,
    allowSorting: true,
    columnMenuOpen: function (args: ColumnMenuOpenEventArgs) {
        for (let item of args.items) {
            if (item.text === 'Filter' && args.column.field === 'OrderID') {
                (item as ColumnMenuItemModel).hide = true;
            } else {
                (item as ColumnMenuItemModel).hide = false;
            }
        }
    },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 200, textAlign:
'Right', showInColumnChooser: false },
        { field: 'Freight', width: 150, format: 'C2', textAlign: 'Right',
editType: 'numericedit' },
        { field: 'ShipName', headerText: 'Ship Name', width: 300 },
        { field: 'ShipCountry', headerText: 'Ship Country', visible: false,
width: 200 },
        { field: 'ShipCity', headerText: 'Ship City', width: 200 }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>

```

```

        <td>${FirstName} </td>
    </tr>
    <tr>
        <td class="CardHeader">Last Name</td>
        <td>${LastName} </td>
    </tr>
    <tr>
        <td class="CardHeader">Title
        </td>
        <td>${Title}
        </td>
    </tr>
    <tr>
        <td class="CardHeader">Country
        </td>
        <td>${Country}
        </td>
    </tr>
</tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the icon of column menu

You can customize the column menu icon by overriding the default grid class `.e-icons.e-columnmenu` with a custom property `content` as mentioned below.

```

,

.e-grid .e-columnheader .e-icons.e-columnmenu::before {
content: "\e941";
}
,

```

In the below sample, grid is rendered with a customized column menu icon.

INDEX.TS

```

import { Grid, ColumnMenu } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(ColumnMenu);
let grid: Grid = new Grid({

```

```

dataSource: data,
columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipName', headerText: 'Ship Name', width: 140 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
],
showColumnMenu: true,
height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

  <style>
    .e-grid .e-columnheader .e-icons.e-columnmenu::before {
      content: "\e941";
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column spanning in EJ2 JavaScript Grid control

The column spanning feature in the Syncfusion Grid allows you to merge adjacent cells horizontally, creating a visually appealing and informative layout. By defining the [colSpan](#) attribute in the [queryCellInfo](#) event, you can easily span cells and customize the appearance of the grid.

In the following demo, Employee **Davolio** doing analysis from 9.00 AM to 10.00 AM, so that cells have spanned.

INDEX.TS

```

import { Grid, QueryCellInfoEventArgs } from '@syncfusion/ej2-grids';
import { columnSpanData, ColumnSpanDataType } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: columnSpanData,
    queryCellInfo: QueryCellEvent,
    gridLines: 'Both',
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', isPrimaryKey:
true, textAlign: 'Right', width: 150 },
        { field: 'EmployeeName', headerText: 'Employee Name', width: 200 },
        { field: '9:00', headerText: '9.00 AM', width: 120 },
        { field: '9:30', headerText: '9.30 AM', width: 120 },
        { field: '10:00', headerText: '10.00 AM', width: 120 },
        { field: '10:30', headerText: '10.30 AM', width: 120 },
        { field: '11:00', headerText: '11.00 AM', width: 120 },
        { field: '11:30', headerText: '11.30 AM', width: 120 },
        { field: '12:00', headerText: '12.00 PM', width: 120 },
        { field: '12:30', headerText: '12.30 PM', width: 120 },
        { field: '2:30', headerText: '2.30 PM', width: 120 },
        { field: '3:00', headerText: '3.00 PM', width: 120 },
        { field: '3:30', headerText: '3.30 PM', width: 120 },
        { field: '4:00', headerText: '4.00 PM', width: 120 },
        { field: '4:30', headerText: '4.30 PM', width: 120 },
        { field: '5:00', headerText: '5.00 PM', width: 120 }
    ],
    width: 'auto',
    height: 'auto',
    allowTextWrap: true
});

```



```

grid.appendTo('#Grid');
function QueryCellEvent(args: QueryCellInfoEventArgs): void {
    let data: ColumnSpanDataType = args.data as ColumnSpanDataType;
    switch (data.EmployeeID) {
        case 10001:
            if (args.column.field === '9:00' || args.column.field === '2:30'
|| args.column.field === '4:30') {
                args.colSpan = 2;
            } else if (args.column.field === '11:00') {
                args.colSpan = 3;
            }
            break;
        case 10002:
            if (args.column.field === '9:30' || args.column.field === '2:30'
||
                args.column.field === '4:30') {
                args.colSpan = 3;
            } else if (args.column.field === '11:00') {
                args.colSpan = 4;
            }
            break;
        case 10003:
            if (args.column.field === '9:00' || args.column.field ===
'11:30') {
                args.colSpan = 3;
            } else if (args.column.field === '10:30' || args.column.field
=== '3:30' ||
                args.column.field === '4:30' || args.column.field ===
'2:30') {
                args.colSpan = 2;
            }
            break;
        case 10004:
            if (args.column.field === '9:00') {
                args.colSpan = 3;
            } else if (args.column.field === '11:00') {
                args.colSpan = 4;
            } else if (args.column.field === '4:00' || args.column.field ===
'2:30') {
                args.colSpan = 2;
            }
            break;
        case 10005:
            if (args.column.field === '9:00') {
                args.colSpan = 4;
            } else if (args.column.field === '11:30') {
                args.colSpan = 3;
            } else if (args.column.field === '3:30' || args.column.field ===
'4:30' || args.column.field === '2:30') {
                args.colSpan = 2;
            }
            break;
        case 10006:
            if (args.column.field === '9:00' || args.column.field === '4:30'
||
                args.column.field === '2:30' || args.column.field ===
'3:30') {

```

```

        args.colSpan = 2;
    } else if (args.column.field === '10:00' || args.column.field
=== '11:30') {
        args.colSpan = 3;
    }
    break;
    case 10007:
        if (args.column.field === '9:00' || args.column.field === '3:00'
|| args.column.field === '10:30') {
            args.colSpan = 2;
        } else if (args.column.field === '11:30' || args.column.field
=== '4:00') {
            args.colSpan = 3;
        }
        break;
    case 10008:
        if (args.column.field === '9:00' || args.column.field ===
'10:30' || args.column.field === '2:30') {
            args.colSpan = 3;
        } else if (args.column.field === '4:00') {
            args.colSpan = 2;
        }
        break;
    case 10009:
        if (args.column.field === '9:00' || args.column.field ===
'11:30') {
            args.colSpan = 3;
        } else if (args.column.field === '4:30' || args.column.field ===
'2:30') {
            args.colSpan = 2;
        }
        break;
    case 10010:
        if (args.column.field === '9:00' || args.column.field === '2:30'
||
        args.column.field === '4:00' || args.column.field ===
'11:30') {
            args.colSpan = 3;
        } else if (args.column.field === '10:30') {
            args.colSpan = 2;
        }
        break;
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change the border color while column spanning

You can change the border color for the spanned cells by the using [queryCellInfo](#) event. This event triggers before the cell element is appended to the Grid element.

INDEX.TS

```

import { Grid, QueryCellInfoEventArgs } from '@syncfusion/ej2-grids';
import { columnSpanData, ColumnSpanDataType } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: columnSpanData,
  queryCellInfo: QueryCellEvent,
  gridLines: 'Both',
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', isPrimaryKey:
true, textAlign: 'Right', width: 150 },
    { field: 'EmployeeName', headerText: 'Employee Name', width: 200 },
    { field: '9:00', headerText: '9.00 AM', width: 120 },

```

```

        { field: '9:30', headerText: '9.30 AM', width: 120 },
        { field: '10:00', headerText: '10.00 AM', width: 120 },
        { field: '10:30', headerText: '10.30 AM', width: 120 },
        { field: '11:00', headerText: '11.00 AM', width: 120 },
        { field: '11:30', headerText: '11.30 AM', width: 120 },
        { field: '12:00', headerText: '12.00 PM', width: 120 },
        { field: '12:30', headerText: '12.30 PM', width: 120 },
        { field: '2:30', headerText: '2.30 PM', width: 120 },
        { field: '3:00', headerText: '3.00 PM', width: 120 },
        { field: '3:30', headerText: '3.30 PM', width: 120 },
        { field: '4:00', headerText: '4.00 PM', width: 120 },
        { field: '4:30', headerText: '4.30 PM', width: 120 },
        { field: '5:00', headerText: '5.00 PM', width: 120 }
    ],
    width: 'auto',
    height: 'auto',
    allowTextWrap: true
});
grid.appendTo('#Grid');
function QueryCellEvent(args: QueryCellInfoEventArgs): void {
    let data: ColumnSpanDataType = args.data as ColumnSpanDataType;
    switch (data.EmployeeID) {
        case 10001:
            if (
                args.column.field === '9:00' ||
                args.column.field === '2:30' ||
                args.column.field === '4:30'
            ) {
                args.colSpan = 2;
            } else if (args.column.field === '11:00') {
                args.colSpan = 3;
            }
            break;
        case 10002:
            if (
                args.column.field === '9:30' ||
                args.column.field === '2:30' ||
                args.column.field === '4:30'
            ) {
                args.colSpan = 3;
            } else if (args.column.field === '11:00') {
                args.colSpan = 4;
            }
            break;
        case 10003:
            if (args.column.field === '9:00' || args.column.field === '11:30') {
                args.colSpan = 3;
            } else if (
                args.column.field === '10:30' ||
                args.column.field === '3:30' ||
                args.column.field === '4:30' ||
                args.column.field === '2:30'
            ) {
                args.colSpan = 2;
            }
            break;
        case 10004:

```

```

        if (args.column.field === '9:00') {
            args.colSpan = 3;
        } else if (args.column.field === '11:00') {
            args.colSpan = 4;
        } else if (args.column.field === '4:00' || args.column.field ===
'2:30') {
            args.colSpan = 2;
        }
        break;
    case 10005:
        if (args.column.field === '9:00') {
            args.colSpan = 4;
        } else if (args.column.field === '11:30') {
            args.colSpan = 3;
        } else if (
            args.column.field === '3:30' ||
            args.column.field === '4:30' ||
            args.column.field === '2:30'
        ) {
            args.colSpan = 2;
        }
        break;
    case 10006:
        if (
            args.column.field === '9:00' ||
            args.column.field === '4:30' ||
            args.column.field === '2:30' ||
            args.column.field === '3:30'
        ) {
            args.colSpan = 2;
        } else if (
            args.column.field === '10:00' ||
            args.column.field === '11:30'
        ) {
            args.colSpan = 3;
        }
        break;
    case 10007:
        if (
            args.column.field === '9:00' ||
            args.column.field === '3:00' ||
            args.column.field === '10:30'
        ) {
            args.colSpan = 2;
        } else if (
            args.column.field === '11:30' ||
            args.column.field === '4:00'
        ) {
            args.colSpan = 3;
        }
        break;
    case 10008:
        if (
            args.column.field === '9:00' ||
            args.column.field === '10:30' ||
            args.column.field === '2:30'
        ) {

```

```

        args.colSpan = 3;
    } else if (args.column.field === '4:00') {
        args.colSpan = 2;
    }
    break;
default:
    extendQueryCellEvent(args, data.EmployeeID);
}
if (args.colSpan > 1) {
    (args.cell as HTMLElement).style.border = '1px solid red';
}
}

function extendQueryCellEvent(
    args: QueryCellInfoEventArgs,
    value: number
): void {
    switch (value) {
        case 10009:
            if (args.column.field === '9:00' || args.column.field === '11:30') {
                args.colSpan = 3;
            } else if (args.column.field === '4:30' || args.column.field ===
'2:30') {
                args.colSpan = 2;
            }
            break;
        case 10010:
            if (
                args.column.field === '9:00' ||
                args.column.field === '2:30' ||
                args.column.field === '4:00' ||
                args.column.field === '11:30'
            ) {
                args.colSpan = 3;
            } else if (args.column.field === '10:30') {
                args.colSpan = 2;
            }
            break;
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations

- Column spanning is not compatible with the following features:
 1. Virtual scrolling
 2. Infinite scrolling
 3. Lazy load grouping

Responsive columns in EJ2 JavaScript Grid control

You can toggle column visibility based on media queries which are defined at the [hideAtMedia](#).

The [hideAtMedia](#) accepts valid [Media Queries](#). In the below sample, for **OrderID** column, [hideAtMedia](#) property value is set as (**min-width: 700px**) so that **OrderID** column will gets hidden when the browser screen width is less than 700px.

INDEX.TS

```

import { Grid, Selection } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Selection);
let grid: Grid = new Grid({

```

```

    dataSource: data,
    columns: [
        {
            field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right',
            hideAtMedia: '(min-width: 700px) '
        }, // column hides when browser screen width less than 700px;
        {
            field: 'CustomerID', headerText: 'Customer ID', width: 150,
            hideAtMedia: '(max-width: 500px) '
        }, // column shows when browser screen width less than or equal to
500px;
        {
            field: 'OrderDate', headerText: 'Order Date', width: 130,
format: 'yMd',
            textAlign: 'Right', hideAtMedia: '(min-width: 500px) '
        }, // column hides when browser screen size less than 500px;
        { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right' }
    ], // it always shown
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Row

Row in EJ2 JavaScript Grid control

The row represents record details fetched from data source.

Row customization

Using event

You can customize the appearance of a row by using the [rowDataBound](#) event. The [rowDataBound](#) event triggers for every row. In the event handler, you can get the

[RowDataBoundEventArgs](#) that contains details of the row.

INDEX.TS

```

import { Grid, RowDataBoundEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    enableHover: false,
    allowSelection: false,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'Freight', headerText: 'Freight', width: 100, format: 'C2'
},
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    rowDataBound: rowBound,
    height: 280
});

```

```

grid.appendTo('#Grid');
function rowBound(args: RowDataBoundEventArgs) {
    if (args.data['Freight'] < 30) {
        args.row.classList.add('below-30');
    } else if (args.data['Freight'] < 80) {
        args.row.classList.add('below-80');
    } else {
        args.row.classList.add('above-80');
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using CSS customize alternate rows

You can change the grid's alternative rows' background color by overriding the **.e-altrow** class.

```

.e-grid .e-altrow {
background-color: #fafafa;
}

```

Please refer to the following example.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data.slice(0, 8),
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd' }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using CSS customize selected row

The background color of the selected row can be changed by overriding the following CSS style.

```

.e-grid td.e-active {
background-color: #f9920b;
}

```

This is demonstrated in the following sample:

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data.slice(0, 8),
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd' }
    ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding a new row programmatically

The Grid can add a new row between the existing rows using the [addRecord](#) method of the Grid.

This is demonstrated in the following sample:

INDEX.TS

```

import { Grid, Edit } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit);
let grid: Grid = new Grid({
    dataSource: data,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'Freight', headerText: 'Freight', width: 100, format: 'C2'
},
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ]
});
grid.appendTo('#Grid');
document.getElementById('addrow').onclick = () => {
    grid.addRecord(
        { OrderID: 3232, CustomerID: 'ALKIT', ShipCity: 'London', Freight: 40,
ShipName: 'Que Delícia'}, 2);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="addrow">Add Row</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

When working with remote data, it is impossible to add a new row between the existing rows.

How to get the row information when hovering over the cell

It is possible to get the row information when hovering over the specific cell. This can be achieved by using the [rowDataBound](#) event and [getRowInfo](#) method of the Grid.

In the following sample, the `mouseover` event is bound to a grid row in the `rowDataBound` event, and when hovering over the specific cell, using the `getRowInfo` method, row information will be retrieved and displayed in the console.

INDEX.TS

```
import { Grid, Edit } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { MouseEventArgs } from '@syncfusion/ej2-base';
Grid.Inject(Edit);
let grid: Grid = new Grid({
    dataSource: data,
    rowDataBound: rowDataBound,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'Freight', headerText: 'Freight', width: 100, format: 'C2'
},
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ]
});
grid.appendTo('#Grid');
function rowDataBound(args) {
    args.row.addEventListener('mouseover', function(e) {
        console.log(grid.getRowInfo(e.target))
    })
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to get Data table row value on onclick event using JavaScript?](#)
- [How to maintain selected rows after adding new record](#)
- [How to select the specific record in the grid using its primary key value](#)
- [How to achieve drag and drop the rows in Grid with custom data binding](#)
- [How to get selected records on custom toolbar click](#)

Row height in EJ2 JavaScript Grid control

You can customize the row height of grid rows through the [rowHeight](#) property. The **rowHeight** property

is used to change the row height of entire grid rows.

In the below example, the **rowHeight** is set as '60px'.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data, RowDataBoundEventArgs } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data.slice(0, 8),
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID',
type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ],
  rowHeight: 60
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize row height for particular row

Grid row height for particular row can be customized using the [rowDataBound](#) event by setting the **rowHeight** in arguments for each row based on the requirement.

In the below example, the row height for the row with OrderID as '10249' is set as '90px' using the **rowDataBound** event.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { Query, DataManager } from '@syncfusion/ej2-data';
import { data, RowDataBoundEventArgs } from './datasource.ts';
let gridData: Object = new DataManager(data).executeLocal(new
Query().take(8));
let grid: Grid = new Grid({
    dataSource: gridData,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },

```

```

        { field: 'CustomerID', width: 140, headerText: 'Customer ID',
type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ],
    rowDataBound: rowDataBound
});
function rowDataBound(args: RowDataBoundEventArgs) {
    if((args.data as OrderDetails).OrderID === 10249){
        args.rowHeight = 90;
    }
}
interface OrderDetails {
    OrderID?: number;
}
grid.appendTo('#Grid');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* In virtual scrolling mode, it is not applicable to set the **rowHeight** using the **rowDataBound** event.

Row template in EJ2 JavaScript Grid control

The row template feature in Grid allows you to customize the appearance and layout of rows in the grid. This feature is useful when you want to display custom content, such as images, buttons, or other controls, within the rows.

To enable the row template feature, you need to set the [rowTemplate](#) property of the Grid control. This property accepts a custom HTML template that defines the layout for each row.

In the following example, Employee Information with Employee Photo is presented in the first column and employee details like Name, Address, etc., are presented in the second column.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { employeeData } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: employeeData,
    rowTemplate: '#rowtemplate',
    columns: [
        { headerText: 'Employee Image', width: 150, textAlign: 'Center',
field: 'OrderID' },
        { headerText: 'Employee Details', width: 300, field: 'EmployeeID' }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

```

```

        </table>
    </td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Row template with formatting

The row template feature in Syncfusion Grid allows you to customize the layout of rows in the grid. This is useful when you want to display images, buttons, or other custom content within the rows of a grid.

By default, Syncfusion Grid provides the [columns->format](#) property to format the values displayed in each column. However, when using the [rowtemplate](#), the [columns->format](#) property cannot be directly applied to format the values inside the template.

To format the values within the row template, you can define a global function that handles the formatting logic. This function can be invoked inside the template to format the corresponding values.

Here is an example of how to define a global formatting function for a date column and use it inside a [rowTemplate](#):

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { employeeData } from './datasource.ts';
import { Internationalization } from '@syncfusion/ej2-base';
let intl: Internationalization = new Internationalization();
let dFormatter: Function = intl.getDateFormat({ skeleton: 'yMd', type:
'date' });
(<IWindow>window).formatDate = (date: Date) => dFormatter(date);
let grid: Grid = new Grid({
    dataSource: employeeData,
    rowTemplate: '#rowtemplate',
    columns: [
        { headerText: 'Employee Image', width: 150, textAlign: 'Center',
field: 'OrderID' },
        { headerText: 'Employee Details', width: 300, field: 'EmployeeID' }
    ],
    height: 315
});
grid.appendTo('#Grid');
interface IWindow extends Window {
    formatDate?: Function;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

```



```

        <tr>
            <td class="CardHeader">Title
            </td>
            <td>${Title}
            </td>
        </tr>
        <tr>
            <td class="CardHeader">Birth Date
            </td>
            <td>${formatDate(data.BirthDate)}
            </td>
        </tr>
        <tr>
            <td class="CardHeader">Hire Date
            </td>
            <td>${formatDate(data.HireDate)}
            </td>
        </tr>
    </tbody>
</table>
</td>
</tr>
</script>
<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

When using the `rowTemplate` feature in Syncfusion Grid, keep in mind that any formatting applied to columns using the `columns->format` property will not work inside the template.

Render syncfusion control in row template

The Grid allows you to render custom Syncfusion controls within the rows of the grid. This feature is helpful as it enables you to display interactive Syncfusion controls instead of field values in the grid.

To enable a Syncfusion control in a row template, you need to set the [rowTemplate](#) property of the Grid control. This property accepts a custom HTML template that defines the layout for each row.

Here is an example that demonstrates rendering Syncfusion controls within a row template :

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { ChipList } from '@syncfusion/ej2-buttons';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { DatePicker } from '@syncfusion/ej2-calendars';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
let grid: Grid = new Grid({

```

```

dataSource: data,
rowTemplate: '#rowtemplate',
columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120 },
    { field: 'Quantity', headerText: 'Quantity', width: 170 },
    { field: 'ShipAddress', headerText: 'Ship Address', width: 170 },
    { field: 'OrderDate', headerText: 'Order Date', width: 120 },
    { field: 'OrderStatus', headerText: 'Order Status', width: 120 },
],
dataBound: () => {
    let gridInstance = (document.getElementById('Grid') as
Grid).ej2_instances[0];
    let chipList =
gridInstance.getContentTable().querySelectorAll('.chipList');
    for (let i = 0; i < chipList.length; i++) {
        let chipValue = chipList[i].innerText;
        new ChipList({ chips: [chipValue] }, chipList[i]);
    }
    let NumericList =
gridInstance.getContentTable().querySelectorAll('.numeric');
    for (let i = 0; i < NumericList.length; i++) {
        let numeric: NumericTextBox = new NumericTextBox({});
        numeric.appendTo(NumericList[i]);
    }
    let dateList =
gridInstance.getContentTable().querySelectorAll('.date-input');
    for (let i = 0; i < dateList.length; i++) {
        let dateInput = dateList[i];
        let dateValue = dateInput.value;
        let datepickerObject: DatePicker = new DatePicker({
            value: new Date(dateValue),
        });
        datepickerObject.appendTo(dateInput);
    }

    let dropdownList =
gridInstance.getContentTable().querySelectorAll('.dropdownlist-input');
    for (let i = 0; i < dropdownList.length; i++) {
        let dropdownInputValue = dropdownList[i];
        let dropData = ['Processing', 'Order Placed', 'Delivered'];
        let dropdown: DropDownList = new DropDownList({
            dataSource: dropData,
            value: dropdownList[i].value,
            popupHeight: 150,
            popupWidth: 150
        });
        dropdown.appendTo(dropdownInputValue);
    }
},
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">

```

```

<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr class="rows">
      <td class="chipList"> ${OrderID}</td>
      <td><input class="numeric" type="text" value=${Quantity}></td>
      <td>${ShipAddress} </td>
      <td><input class="date-input" value="${OrderDate}"></td>
      <td><input class="dropdownlist-input" type="text"
value="${OrderStatus}"></td>
    </tr>
  </script>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Limitations

Row template feature is not compatible with all the features which are available in the grid, and it has limited features support. The features that are incompatible with the row template feature are listed below.

- Filtering
- Paging
- Sorting
- Searching
- Rtl
- Export
- Context Menu
- State Persistence
- Selection
- Grouping
- Editing
- Frozen rows & columns
- Virtual & Infinite scrolling
- Column chooser
- Column menu
- Detail Row
- Foreignkey column
- Resizing
- Reordering
- Aggregates
- Clipboard
- Adaptive view

Detail template in EJ2 JavaScript Grid control

The detail template in the Grid control allows you to display additional information about a specific row in the grid by expanding or collapsing detail content. This feature is useful when you need to show additional data or custom content that is specific to each row in the grid. You can use the [detailTemplate](#) property to define an HTML template for the detail row. This template can include any HTML element or EJ2 JavaScript control that you want to display as detail content.

Here's an example of using the `detailTemplate` property in the grid control:

INDEX.TS

```
import { Grid, DetailRow } from '@syncfusion/ej2-grids';
import { employeeData } from './datasource.ts';
Grid.Inject(DetailRow);
let grid: Grid = new Grid({
  dataSource: employeeData,
  detailTemplate: '#detailtemplate',
  columns: [
    { field: 'FirstName', headerText: 'First Name', width: 140 },
    { field: 'LastName', headerText: 'Last Name', width: 140 },
    { field: 'Title', headerText: 'Title', width: 150 },
```

```

        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script id="detailtemplate" type="text/x-template">
        <table class="detailtable" width="100%" >
            <colgroup>
                <col width="35%">
                <col width="35%">
                <col width="30%">
            </colgroup>
            <tbody>
                <tr>
                    <td rowspan="4" style="text-align: center;">
                        <img class='photo' src='${EmployeeID}.png'
alt='${EmployeeID}' />

```

```

        </td>
        <td>
            <span style="font-weight: 500;">First Name: </span>
${FirstName}
        </td>
        <td>
            <span style="font-weight: 500;">Postal Code: </span>
${PostalCode}
        </td>
    </tr>
    <tr>
        <td>
            <span style="font-weight: 500;">Last Name: </span>
${LastName}
        </td>
        <td>
            <span style="font-weight: 500;">City: </span> ${City}
        </td>
    </tr>
    <tr>
        <td>
            <span style="font-weight: 500;">Title: </span> ${Title}
        </td>
        <td>
            <span style="font-weight: 500;">Phone: </span>
${HomePhone}
        </td>
    </tr>
    <tr>
        <td>
            <span style="font-weight: 500;">City: </span> ${City}
        </td>
        <td>
            <span style="font-weight: 500;">Country: </span>
${Country}
        </td>
    </tr>
</tbody>
</table>
</script>
<div id="container">
    <div id="Grid"></div>
</div>
<script>
    var ele = document.getElementById('container');
    if (ele) {
        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Rendering custom control

The Grid control provides a powerful feature that allows you to render custom controls inside the detail row. This feature is helpful when you need to add additional information or functionality for a specific row in the grid.

To render a custom control inside the detail row, you need to define a template using the [detailTemplate](#) property and handle the [detailDataBound](#) event. This template can include any HTML element or EJ2 JavaScript control that you want to display as the detail content.

The [detailDataBound](#) event is an event that is triggered after a detail row is bound to data. This event provides an object of type [DetailDataBoundEventArgs](#) as a parameter.

For example, to render grid inside the detail row, place an HTML div element as the [detailTemplate](#) and render the DIV element as grid control in the [detailDataBound](#) event.

INDEX.TS

```
import { Grid, DetailRow, DetailDataBoundEventArgs } from '@syncfusion/ej2-grids';
import { employeeData, data } from './datasource.ts';
Grid.Inject(DetailRow);
let grid: Grid = new Grid({
  dataSource: employeeData,
  detailTemplate: '#detailtemplate',
  columns: [
    { field: 'FirstName', headerText: 'First Name', width: 140 },
    { field: 'LastName', headerText: 'Last Name', width: 140 },
    { field: 'Title', headerText: 'Title', width: 150 },
    { field: 'Country', headerText: 'Country', width: 150 }
  ],
  detailDataBound: (e: DetailDataBoundEventArgs) => {
    let detail = new Grid({
      dataSource: data.filter((item: Object) => item['EmployeeID'] === e.data['EmployeeID']),
      columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 110 },
        { field: 'CustomerID', headerText: 'Customer Name', width: 140 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
      ]
    });
    detail.appendTo(<HTMLElement>e.detailElement.querySelector('.custom-grid'));
  }
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="detailtemplate" type="text/x-template">
    <div class='custom-grid'></div>
  </script>
  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Expand by external button

The Grid provides a feature that allows users to expand the detail row of a grid using an external button. By default, detail rows render in a collapsed state, but this feature enables users to view additional details associated with a particular row.

To achieve expanding the detail row of a grid using an external button, you need to invoke the [expand](#) method provided by the **detailRowModule** object of the Syncfusion Grid library. This method will expand the detail row of a specific grid row.

Here is an example of how to use the `expand` method to expand a detail row:

INDEX.TS


```

import { Grid, DetailRow } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { employeeData } from './datasource.ts';
import { TextBox } from '@syncfusion/ej2-inputs';
Grid.Inject(DetailRow);
let grid: Grid = new Grid({
    dataSource: employeeData,
    detailTemplate: '#detailtemplate',
    columns: [
        { field: 'FirstName', headerText: 'First Name', width: 140 },
        { field: 'LastName', headerText: 'Last Name', width: 140 },
        { field: 'Title', headerText: 'Title', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    height: 260
});
grid.appendTo('#Grid');
let expandButton: Button = new Button();
expandButton.appendTo('#expand');
let textbox: TextBox = new TextBox({
    placeholder: 'Enter the Row Index',
    floatLabelType: 'Auto',
    width: 250
});
textbox.appendTo('#rowindex');
(document.getElementById('expand') as HTMLElement).addEventListener('click',
() => {
    grid.detailRowModule.expand(textbox.value);
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<script id="detailtemplate" type="text/x-template">
<table class="detailtable" width="100%" >
<colgroup>
<col width="35%">
<col width="35%">
<col width="30%">
</colgroup>
<tbody>
<tr>
<td rowspan="4" style="text-align: center;">

</td>
<td>
<span style="font-weight: 500;">First Name: </span>
${FirstName}
</td>
<td>
<span style="font-weight: 500;">Postal Code: </span>
${PostalCode}
</td>
</tr>
<tr>
<td>
<span style="font-weight: 500;">Last Name: </span>
${LastName}
</td>
<td>
<span style="font-weight: 500;">City: </span> ${City}
</td>
</tr>
<tr>
<td>
<span style="font-weight: 500;">Title: </span> ${Title}
</td>
<td>
<span style="font-weight: 500;">Phone: </span>
${HomePhone}
</td>
</tr>
<tr>
<td>
<span style="font-weight: 500;">City: </span> ${City}
</td>
<td>

```

```

        <span style="font-weight: 500;">Country: </span>
    </td>
</tr>
</tbody>
</table>
</script>
<div id="container">
    <div class="e-float-input" style="width: 250px; display: inline-block ;padding: 0px 30px 0px 0px">
        <input id="rowindex" type="text" />
        <span class="e-float-line"></span>
    </div>
    <button id="expand">Expand</button>
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize detail template icon

The detail template icon in the Syncfusion Grid is used to expand or collapse the detail content of a row. By default, the icon represents a right arrow for the collapsed state and a down arrow for the expanded state. If you want to customize this icon, you can achieve it by overriding the following CSS styles:

```

`css
.e-grid .e-icon-grightarrow::before {
content: '\e300';
}
.e-grid .e-icon-gdownarrow::before {
content: '\e304';
}
`

```

Here is an example of how to customize the detail template icon:

INDEX.TS

```

import { Grid, DetailRow } from '@syncfusion/ej2-grids';
import { employeeData } from './datasource.ts';
Grid.Inject(DetailRow);
let grid: Grid = new Grid({
    dataSource: employeeData,
    detailTemplate: '#detailtemplate',
    columns: [
        { field: 'FirstName', headerText: 'First Name', width: 140 },

```

```

        { field: 'LastName', headerText: 'Last Name', width: 140 },
        { field: 'Title', headerText: 'Title', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <style>
        .e-grid .e-icon-grightarrow::before {
            content: '\e300';
        }
        .e-grid .e-icon-gdownarrow::before {
            content: '\e304';
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script id="detailtemplate" type="text/x-template">
        <table class="detailtable" width="100%" >
            <colgroup>

```

```

        <col width="35%">
        <col width="35%">
        <col width="30%">
    </colgroup>
    <tbody>
    <tr>
        <td rowspan="4" style="text-align: center;">
            
        </td>
        <td>
            <span style="font-weight: 500;">First Name: </span>
            {FirstName}
        </td>
        <td>
            <span style="font-weight: 500;">Postal Code: </span>
            {PostalCode}
        </td>
    </tr>
    <tr>
        <td>
            <span style="font-weight: 500;">Last Name: </span>
            {LastName}
        </td>
        <td>
            <span style="font-weight: 500;">City: </span> {City}
        </td>
    </tr>
    <tr>
        <td>
            <span style="font-weight: 500;">Title: </span> {Title}
        </td>
        <td>
            <span style="font-weight: 500;">Phone: </span>
            {HomePhone}
        </td>
    </tr>
    <tr>
        <td>
            <span style="font-weight: 500;">City: </span> {City}
        </td>
        <td>
            <span style="font-weight: 500;">Country: </span>
            {Country}
        </td>
    </tr>
    </tbody>
</table>
</script>
<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Limitations

Detail template is not supported with the following features:

- Frozen rows and columns
- Immutable mode
- Infinite scrolling
- Virtual scrolling
- Print
- Row template
- Row spanning
- Column spanning
- Lazy load grouping
- State persistence

Row drag and drop in EJ2 JavaScript Grid control

The Syncfusion EJ2 JavaScript Grid control provides built-in support for row drag and drop functionality. This feature allows you to easily rearrange rows within the grid by dragging and dropping them to new positions. Additionally, you can also drag and drop rows from one grid to another grid, as well as drag and drop rows to custom controls.

To use the row drag and drop feature in Grid control, you need to inject the **RowDD** module in the grid. The **RowDD** is responsible for handling the row drag and drop functionality in the grid control. Once you have injected the **RowDD**, you can then use the [allowRowDragAndDrop](#) and [targetID](#) properties to enable and configure the row drag and drop feature in the Grid.

Drag and drop within grid

The drag and drop feature allows you to rearrange rows within the grid by dragging them using a drag icon. This feature can be enabled by setting the [allowRowDragAndDrop](#) property to **true**. This property is a boolean value that determines whether row drag and drop is enabled or not. By default, it is set to **false**, which means that row drag and drop is disabled.

Here's an example of how to enable drag and drop within the Grid:

INDEX.TS

```
import { Grid, RowDD, Selection } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(RowDD, Selection);
let grid: Grid = new Grid({
  dataSource: data,
  allowRowDragAndDrop: true,
  selectionSettings: { type: 'Multiple' },
  height: 400,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign: 'Right' },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
  ]
});
```

```

        { field: 'OrderDate', headerText: 'Order Date', width: 100,
format:'yMd', textAlign: 'Right' },
        { field: 'Freight', headerText: 'Freight', width: 120, format:'C2',
textAlign: 'Right' },
        { field: 'ShipCity', headerText: 'Ship City', width: 130 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 130 }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div style="display: inline-block">
            <div id="Grid"></div>
        </div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop to grid

The grid row drag and drop allows you to drag grid rows and drop to another grid. This feature can be enabled by setting the [allowRowDragAndDrop](#) property to **true** in the Grid control. This property specifies whether to enable or disable the row drag and drop feature in the Grid. By default, this property is set to **false**, which means that row drag and drop functionality is not enabled.

To specify the target control where the grid rows should be dropped, use the [targetID](#) property of the [rowDropSettings](#) object. The [targetID](#) property takes the ID of the target control as its value.

Here's an example code snippet that demonstrates how to enable Row drag and drop another Grid control:

INDEX.TS

```

import { Grid, RowDD, Selection } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject( RowDD, Selection);
let grid: Grid = new Grid({
    dataSource: data,
    allowRowDragAndDrop: true,
    rowDropSettings: { targetID: 'DestGrid' },
    selectionSettings: { type: 'Multiple' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
});
grid.appendTo('#Grid');
let destGrid: Grid = new Grid({
    dataSource: [],
    allowRowDragAndDrop: true,
    rowDropSettings: { targetID: 'Grid' },
    selectionSettings: { type: 'Multiple' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
});
destGrid.appendTo('#DestGrid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>

```



```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div style="display: inline-block">
      <div id="Grid"></div>
      <div id="DestGrid"></div>
    </div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The row drag and drop feature is not supported in virtual scrolling and frozen rows and columns mode.

* In order to use row drag and drop, you need to inject the **RowDD** module in the grid.

Drag and drop to custom control

The Grid provides the feature to drag and drop grid rows to any custom control. This feature allows you to easily move rows from one control to another without having to manually copy and paste data. To enable row drag and drop, you need to set the [allowRowDragAndDrop](#) property to **true** and defining the

custom control ID in the [targetID](#) property of the `rowDropSettings` object. The ID provided in `targetID` should correspond to the ID of the target control where the rows are to be dropped.

In the below example, the selected grid row is dragged and dropped in to the TreeGrid control by using [rowDrop](#) event. Once the row is dropped into the TreeGrid control, we have removed the corresponding grid row from grid and its data inserted in custom control.

INDEX.TS

```
import { Grid, Page, RowDD, Selection, Edit, RowDragEventArgs } from '@syncfusion/ej2-grids';
import { TreeGrid, Edit as TreeEdit } from '@syncfusion/ej2-treegrid';
import { isNullOrUndefined } from '@syncfusion/ej2-base';
import { sampleGridData } from './datasource.ts';
Grid.Inject(Page, RowDD, Selection, Edit);
TreeGrid.Inject(TreeEdit);
let grid: Grid = new Grid({
  dataSource: sampleGridData,
  allowPaging: true,
  allowSelection: true,
  allowRowDragAndDrop: true,
  rowDropSettings: { targetID: 'TreeGrid' },
  selectionSettings: { type: 'Multiple' },
  editSettings: { allowDeleting: true },
  columns: [
    { field: 'taskID', headerText: 'taskID', textAlign: 'Right', width: 90 },
    { field: 'taskName', headerText: 'taskName', textAlign: 'Left', width: 180 },
    { field: 'description', headerText: 'description', textAlign: 'Left', width: 180 },
    { field: 'category', headerText: 'category', textAlign: 'Left', width: 180 },
    { field: 'startDate', headerText: 'startDate', textAlign: 'Right', format: 'yMd', width: 120 },
    { field: 'duration', headerText: 'duration', textAlign: 'Right', width: 80 }
  ],
  rowDrop: function (args: RowDragEventArgs) {
    let grid = (document.getElementById('Grid') as Grid).ej2_instances[0];
    let tree = (document.getElementById('TreeGrid') as TreeGrid).ej2_instances[0];
    if (args.target.closest('.e-treegrid')) {
      args.cancel = true;
      let rowIndex = !isNullOrUndefined(args.target.closest('.e-row'))
        ? args.target.closest('.e-row').rowIndex
        : 0;
      for (let i = 0; i < args.data.length; i++) {
        tree.addRecord(args.data[i], rowIndex);
        grid.deleteRecord('taskID', args.data[i]);
      }
    }
  },
});
grid.appendTo('#Grid');
let treeGridObj: TreeGrid = new TreeGrid({
  childMapping: 'subtasks',
```

```

editSettings: { allowAdding: true, allowEditing: true },
columns: [
    { field: 'taskID', headerText: 'Task ID', width: 90, textAlign: 'Right'
},
    { field: 'taskName', headerText: 'Task Name', width: 180, textAlign:
'Left' },
    { field: 'startDate', headerText: 'Start Date', width: 90, textAlign:
'Right', type: 'date', format: 'yMd' },
    { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],
});
treeGridObj.appendTo('#TreeGrid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="http://cdn.syncfusion.com/ej2/ej2-
treegrid/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div style="display: inline-block">
            <div id="Grid"></div>

```

```

        <div id="TreeGrid"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The `rowDrop` event is fired when a row is dropped onto a custom control, regardless of whether the drop is successful or not. You can use the `args.cancel` property to prevent the default action.

Drag and drop rows without drag icon

By default, when performing a drag and drop operation in the Syncfusion Grid, drag icons are displayed. However, in some cases, you may want to hide these drag icons. This can be achieved by setting the `targetID` property of the `rowDropSettings` object to the current Grid's ID.

By setting the `targetID`, the Grid will render without a helper icon for row dragging. You can then customize the drag and drop action by binding to the `rowDrop` event of the Grid. In the `rowDrop` event, you can prevent the default action by setting `args.cancel` to `true`, and reorder the rows using the `reorderRows` method of the Grid.

Here's an example of how to hide the drag and drop icon in the Syncfusion Grid:

INDEX.TS

```

import { Grid, RowDD, Selection, RowDragEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(RowDD, Selection);
let grid: Grid = new Grid({
    dataSource: data,
    allowRowDragAndDrop: true,
    rowDropSettings: { targetID: 'Grid' },
    selectionSettings: { type: 'Multiple' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign: 'Right' },
        { field: 'CustomerID', headerText: 'Customer Name', width: 130 },
        { field: 'OrderDate', headerText: 'Order Date', width: 120, format: 'yMd', textAlign: 'Right' },
        { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2', textAlign: 'Right' },
        { field: 'ShipCity', headerText: 'Ship City', width: 130 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 130 },
    ],
    rowDrop: function(args: RowDragEventArgs) {
        args.cancel = true;
        let value = [];
        for (let index = 0; index < args.rows.length; index++) {
            value.push(args.fromIndex + index);
        }
    }
});

```

```

        grid.reorderRows(value, args.dropIndex);
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div style="display: inline-block">
      <div id="Grid"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The selection feature must be enabled in the Grid to allow users to select rows before performing the drag and drop operation.

* Multiple rows can be selected by clicking and dragging inside the grid. For multiple row selection, the [type](#) property must be set to **Multiple**.

Drag and drop events

The Grid control provides a set of events that are triggered during drag and drop operations on grid rows. These events allow you to customize the drag element, track the progress of the dragging operation, and perform actions when a row is dropped on a target element. The following events are available:

1. [rowDragStartHelper](#): This event is triggered when a click occurs on the drag icon or a grid row. It allows you to customize the drag element based on specific criteria.
2. [rowDragStart](#): This event is triggered when the dragging of a grid row starts.
3. [rowDrag](#): This event is triggered continuously while the grid row is being dragged.
4. [rowDrop](#): This event is triggered when a drag element is dropped onto a target element.

INDEX.TS

```
import { Grid, RowDD, Selection, Page, RowDragEventArgs } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(RowDD, Selection, Page );
let grid: Grid = new Grid({
  dataSource: data,
  allowRowDragAndDrop: true,
  allowPaging: true,
  selectionSettings: { type: 'Multiple' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right' },
    { field: 'CustomerID', headerText: 'Customer ID', width: 130 },
    { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd', textAlign: 'Right' },
    { field: 'Freight', headerText: 'Freight', width: 120, format: 'C2',
textAlign: 'Right' }
  ],
  rowDragStart: function (args: RowDragEventArgs) {
    (document.getElementById('message') as HTMLElement).innerText =
`rowDragStart event triggered`;
    args.cancel = true;
  },
  rowDragStartHelper: function (args: RowDragEventArgs) {
    (document.getElementById('message') as HTMLElement).innerText =
`rowDragStartHelper event triggered`;
    if (args.data[0].OrderID === 10248) {
      args.cancel = true;
    }
  },
  rowDrag: function (args: RowDragEventArgs) {
    (document.getElementById('message') as HTMLElement).innerText =
`rowDrag event triggered`;
    args.rows.forEach((row) => {
      row.classList.add('drag-limit');
```

```

    });
  },
  rowDrop: function (args: RowDragEventArgs) {
    (document.getElementById('message') as HTMLElement).innerText =
    `rowDrop event triggered`;
    let value = [];
    for (let index = 0; index < args.rows.length; index++) {
      value.push(args.fromIndex + index);
    }
    grid.reorderRows(value, args.dropIndex);
  },
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <style>
    .event-message {
      text-align: center;
      color: red;
      font-size: 16px;
    }
    .drag-limit .e-rowcell {
      border: 1px solid red;
    }
  </style>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <p class="event-message" id="message"></p>
    <div id="Grid"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations

- Single row is able to drag and drop in same grid without enable the row selection.
- Row drag and drop feature is not having built in support with row template, detail template, hierarchy grid and grouping features of grid.

See also

[Sorting data in the Syncfusion Grid](#)

[Filtering data in the Syncfusion Grid](#)

Row spanning in EJ2 JavaScript Grid control

The grid provides an option to span row cells, allowing you to merge two or more cells in a row into a single cell. This feature can be useful in scenarios where you want to display information that spans across multiple rows, but want to avoid repeating the same information in each row.

To achieve this, You need to define the [rowSpan](#) attribute to span cells in the [queryCellInfo](#) event. The `rowSpan` attribute is used to specify the number of rows that the current cell should span.

The `queryCellInfo` event is triggered for each cell in the grid, and allows you to customize the cells in the grid. By handling this event, you can set the `rowSpan` attribute for a cell to achieve row spanning.

In the following demo, **Davolio** cell is spanned to two rows in the **EmployeeName** column. Also Grid supports the spanning of rows and columns for same cells. **Lunch Break** cell is spanned to two rows and three columns in the **1:00** column.

INDEX.TS

```

import { Grid, QueryCellInfoEventArgs } from '@syncfusion/ej2-grids';
import { columnSpanData, ColumnSpanDataType } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: columnSpanData,
  queryCellInfo: QueryCellEvent,
  gridLines: 'Both',
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', isPrimaryKey:
true, textAlign: 'Right', width: 150 },
    { field: 'EmployeeName', headerText: 'Employee Name', width: 200 },

```



```

        { field: '9:00', headerText: '9.00 AM', width: 120 },
        { field: '9:30', headerText: '9.30 AM', width: 120 },
        { field: '10:00', headerText: '10.00 AM', width: 120 },
        { field: '10:30', headerText: '10.30 AM', width: 120 },
        { field: '11:00', headerText: '11.00 AM', width: 120 },
        { field: '11:30', headerText: '11.30 AM', width: 120 },
        { field: '12:00', headerText: '12.00 PM', width: 120 },
        { field: '12:30', headerText: '12.30 PM', width: 120 },
        { field: '1:00', headerText: '1.00 PM', width: 120 },
        { field: '1:30', headerText: '1.30 PM', width: 120 },
        { field: '2:00', headerText: '2.00 PM', width: 120 },
        { field: '2:30', headerText: '2.30 PM', width: 120 },
        { field: '3:00', headerText: '3.00 PM', width: 120 },
        { field: '3:30', headerText: '3.30 PM', width: 120 },
        { field: '4:00', headerText: '4.00 PM', width: 120 },
        { field: '4:30', headerText: '4.30 PM', width: 120 },
        { field: '5:00', headerText: '5.00 PM', width: 120 }
    ],
    width: 'auto',
    height: 300,
    allowTextWrap: true
});
grid.appendTo('#Grid');
function QueryCellEvent(args: QueryCellInfoEventArgs): void {
    let data: ColumnSpanDataType = args.data as ColumnSpanDataType;
    switch (data.EmployeeID) {
        case 10001:
            if (args.column.field === '9:00' || args.column.field === '2:30'
|| args.column.field === '4:30') {
                args.colSpan = 2;
            } else if (args.column.field === '11:00') {
                args.colSpan = 3;
            } else if (args.column.field === 'EmployeeName') {
                args.rowSpan = 2;
            } else if (args.column.field === '1:00') {
                args.colSpan = 3;
                args.rowSpan = 2;
            }
            break;
        case 10002:
            if (args.column.field === '9:30' || args.column.field === '2:30'
||
                args.column.field === '4:30') {
                args.colSpan = 3;
            } else if (args.column.field === '11:00') {
                args.colSpan = 4;
            }
            break;
        case 10003:
            if (args.column.field === '9:00' || args.column.field ===
'11:30') {
                args.colSpan = 3;
            } else if (args.column.field === '10:30' || args.column.field
=== '3:30' ||
                args.column.field === '4:30' || args.column.field ===
'2:30') {
                args.colSpan = 2;
            }
    }
}

```

```

    }
    break;
case 10004:
    if (args.column.field === '9:00') {
        args.colSpan = 3;
    } else if (args.column.field === '11:00') {
        args.colSpan = 4;
    } else if (args.column.field === '4:00' || args.column.field ===
'2:30') {
        args.colSpan = 2;
    }
    break;
case 10005:
    if (args.column.field === '9:00') {
        args.colSpan = 4;
    } else if (args.column.field === '11:30') {
        args.colSpan = 3;
    } else if (args.column.field === '3:30' || args.column.field ===
'4:30' || args.column.field === '2:30') {
        args.colSpan = 2;
    }
    break;
case 10006:
    if (args.column.field === '9:00' || args.column.field === '4:30'
||
        args.column.field === '2:30' || args.column.field ===
'3:30') {
        args.colSpan = 2;
    } else if (args.column.field === '10:00' || args.column.field
=== '11:30') {
        args.colSpan = 3;
    }
    break;
case 10007:
    if (args.column.field === '9:00' || args.column.field === '3:00'
|| args.column.field === '10:30') {
        args.colSpan = 2;
    } else if (args.column.field === '11:30' || args.column.field
=== '4:00') {
        args.colSpan = 3;
    }
    break;
case 10008:
    if (args.column.field === '9:00' || args.column.field ===
'10:30' || args.column.field === '2:30') {
        args.colSpan = 3;
    } else if (args.column.field === '4:00') {
        args.colSpan = 2;
    }
    break;
case 10009:
    if (args.column.field === '9:00' || args.column.field ===
'11:30') {
        args.colSpan = 3;
    } else if (args.column.field === '4:30' || args.column.field ===
'2:30') {
        args.colSpan = 2;
    }

```

```

    }
    break;
case 100010:
    if (args.column.field === '9:00' || args.column.field === '2:30'
||
        args.column.field === '4:00' || args.column.field ===
'11:30') {
        args.colSpan = 3;
    } else if (args.column.field === '10:30') {
        args.colSpan = 2;
    }
    break;
}
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en">
<head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container">
        <div id="Grid"></div>
    </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To disable the spanning for particular grid page, you need to use **requestType** from **queryCellInfo** event argument.

The **rowSpan** and **colSpan** attributes can be used together to merge cells both vertically and horizontally.

Limitations

- Row spanning is not compatible with the following features:
 1. Virtual scrolling
 2. Infinite scrolling
 3. Lazy load grouping
 4. Row drag and drop

Cell

Cell in EJ2 JavaScript Grid control

Cell customization

The appearance of cells can be customized by using the [queryCellInfo](#) event. The [queryCellInfo](#) event triggers for every cell. In that event handler, you can get [QueryCellInfoEventArgs](#) that contains the details of the cell.

INDEX.TS

```

import { Grid, QueryCellInfoEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    enableHover: false,
    allowSelection: false,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'OrderDate', headerText: 'Order Date', width: 100, format:
'yMd'},
        { field: 'Freight', headerText: 'Freight', width: 100, format:
'C2'},
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    ],
    height: 315,
    queryCellInfo: customiseCell
});
grid.appendTo('#Grid');
function customiseCell(args: QueryCellInfoEventArgs) {
    if(args.column.field === 'Freight') {

```

```

        if (args.data['Freight'] < 30){
            args.cell.classList.add('below-30');
        } else if (args.data['Freight'] < 80 ) {
            args.cell.classList.add('below-80');
        } else {
            args.cell.classList.add('above-80');
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>

```

```

    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom attributes

You can customize the grid cells by adding a CSS class to the [customAttribute](#) property of the column.

```

`css
.e-attr {
background: '#d7f0f4';
}
`ts
{
field: "ShipCity", headerText: "Ship City", customAttributes: {class: "e-attr"}, width: "120"
}
`

```

In the below example, we have customized the cells of **OrderID** and **ShipCity** columns.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', customAttributes:
{class: "e-attr"}, textAlign: 'Right', width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', customAttributes:
{class: "e-attr"}, width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
.e-attr {
    background: #d7f0f4;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid lines

The [gridLines](#) have option to display cell border and it can be defined by the [gridLines](#) property.

The available modes of grid lines are:

- | Modes | Actions |
- |-----|-----|
- | Both | Displays both the horizontal and vertical grid lines.|
- | None | No grid lines are displayed.|
- | Horizontal | Displays the horizontal grid lines only.|
- | Vertical | Displays the vertical grid lines only.|
- | Default | Displays grid lines based on the theme.|

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  gridLines: 'Both',
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

```

```

        </table>
    </td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the grid renders with **Default** mode.

See Also

- [How to get the clicked Grid cell details](#)
- [How to customize the Grid cell values while exporting](#)

Content in EJ2 JavaScript Grid control

The HTML tags can be displayed in the Grid header and content by enabling the [disableHtmlEncode](#) property.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: '<span> Order ID </span>',
        disableHtmlEncode: true, textAlign: 'Right', width: 140 },
        { field: 'CustomerID', headerText: '<span> Customer ID </span>',
        disableHtmlEncode: false, width: 120 },
        { field: 'OrderDate', headerText: 'Order Date', width: 100, format:
        'yMd' },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Auto wrap in EJ2 JavaScript Grid control

The auto wrap allows the cell content of the grid to wrap to the next line when it exceeds the boundary of the cell width. The Cell Content wrapping works based on the position of white space between words. To enable auto wrap, set the [allowTextWrap](#) property to `true`. You can configure the auto wrap mode by setting the [textWrapSettings.wrapMode](#) property.

There are three types of [wrapMode](#). They are:

- **Both:** Both value is set by default. Auto wrap will be enabled for both the content and the header.
- **Header:** Auto wrap will be enabled only for the header.
- **Content:** Auto wrap will be enabled only for the content.

Note: When a column width is not specified, then auto wrap of columns will be adjusted with respect to the grid's width.

In the following example, the [textWrapSettings.wrapMode](#) is set to **Content**.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  gridLines: 'Default',
  allowTextWrap: true,
  textWrapSettings: { wrapMode: 'Content' },
  columns: [
    { field: 'RoolNo', headerText: 'Rool No', width: 120 },
    { field: 'Name', headerText: 'Name of the inventor', width: 100 },
    { field: 'patentfamilies', headerText: 'No of patentfamilies',
width: 100 },
    { field: 'Country', headerText: 'country', width: 130 },
    { field: 'mainfields', headerText: 'Main fields of Invention',
width: 150 },
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clip mode in EJ2 JavaScript Grid control

The clip mode provides options to display its overflow cell content and it can be defined by the [columns.clipMode](#) property.

There are three types of [clipMode](#). They are:

- **Clip:** Truncates the cell content when it overflows its area.
- **Ellipsis:** Displays ellipsis when the cell content overflows its area.
- **EllipsisWithTooltip:** Displays ellipsis when the cell content overflows its area, also it will display the tooltip while hover on ellipsis is applied.

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    columns: [
        { field: 'RoolNo', headerText: 'Rool No', width: 100 },

```

```

    { field: 'Name', headerText: 'Name of the inventor', clipMode:
'Clip', width: 100 },
    { field: 'patentfamilies', headerText: 'No of patent families',
clipMode: 'Ellipsis', width: 100 },
    { field: 'Country', headerText: 'Country', width: 80 },
    { field: 'mainfields', headerText: 'Main fields of Invention',
clipMode: 'EllipsisWithTooltip', width: 100 },
  ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, [columns.clipMode](#) value is **Ellipsis**.

Editing

Edit in EJ2 JavaScript Grid control

The Grid component has options to dynamically insert, delete and update records. Editing feature requires a primary key column for CRUD operations. To define the primary key, set [columns.isPrimaryKey](#) to **true** in particular column.

You can start the edit action either by double clicking the particular row or by selecting the required row and click on **Edit** button in the toolbar. Similarly, you can add a new record to grid either by clicking on **Add** button in the toolbar or on an external button which is bound to invoke the [addRecord](#) method of the grid, **Save** and **Cancel** while in edit mode is possible using respective toolbar icon in grid.

Deletion of the record is possible by selecting the required row and click on **Delete** button in the toolbar.

To use CRUD, inject the [Edit](#) module in grid.

INDEX.TS

```

import { Grid, Edit } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit);
let grid: Grid = new Grid({
    dataSource: data,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
.e-row[aria-selected="true"] .e-customizedExpandcell {
    background-color: #e0e0e0;
}
.e-grid.e-gridhover tr[role='row']:hover {
    background-color: #eee;
}
.e-expand::before {
    content: '\e5b8';
}
.empImage {
    margin: 6px 16px;
    float: left;
    width: 50px;
    height: 50px;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

<div id="container">

```



```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* If [columns.isIdentity](#) is enabled, then it will be considered as a read-only column when editing and adding a record.

* You can disable editing for a particular column, by specifying [columns.allowEditing](#) to `false`.

Toolbar with edit option

The grid toolbar has the [built-in items](#) to execute Editing actions. You can define this by using the [toolbar](#) property.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 265
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Disable editing for particular column

You can disable editing for particular columns by using the [columns.allowEditing](#).

In the following demo, editing is disabled for the **CustomerID** column.

INDEX.TS

```
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120,
allowEditing: false },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 265
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable editing for a particular row or cell

You can disable the editing for a particular row by using the [actionBegin](#) event of Grid based on **requestType** as **beginEdit**.

In the below demo, the rows which are having the value for **ShipCountry** column as "France" is prevented from editing.

INDEX.TS

```

import { Grid, Edit, Toolbar, EditEventArgs, EJ2Intance } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,
  actionBegin: (args: EditEventArgs) => {
    if (args.requestType === 'beginEdit') {
      if (args.rowData.ShipCountry == "France") {
        args.cancel = true;
      }
    }
  },
  toolbar: ['Edit', 'Update', 'Cancel'],
  editSettings: { allowEditing: true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
  ],
  height: 265
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

For batch mode of editing, you can use [cellEdit](#) event of Grid. In the below demo, the cells which are having the value as "France" is prevented from editing.

INDEX.TS

```

import { Grid, Edit, Toolbar, EditEventArgs, EJ2Intance } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,

```

```

    cellEdit: (args: EditEventArgs) => {
        if (args.value == "France") {
            args.cancel = true;
        }
    },
    toolbar: ['Edit', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, mode: 'Batch' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 265
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>

```

```

        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Editing template column

You can edit the template column value by defining the **field** for that particular column.

In the below demo, the **ShipCountry** column is rendered with the template.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },

```



```

    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
      editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', template:
      '#template', editType: 'dropdownedit', width: 150 }
  ],
  height: 265
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <script id="template" type="text/x-template">
      <a href="#">${ShipCountry}</a>
    </script>
  </div>

```

```

        </script>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Troubleshoot editing works only for first row

The Editing functionalities can be performed based upon the primary key value of the selected row. If **primaryKey** is not defined in the grid, then edit or delete action take places the first row.

How to make a Grid column always editable

Make the Grid column always editable using the column template feature of the Grid.

In the following example, the textbox is rendered in the Freight column using a column template. The keyup event for the Grid is bound using the [created](#) event of the Grid, and the edited changes are saved in the data source using the [updateRow](#) method of the Grid.

INDEX.TS

```

import { Grid, parentsUntil } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, isPrimaryKey: true },
        { field: 'OrderDate', headerText: 'Order Date', width: 130,
textAlign: 'Right', format: 'yMd' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 140 },
        { field: 'Freight', headerText: 'Receipt Amount', width: 150,
template: '#template' }
    ],
    height: 315,
    created: (args) => {
        grid.element.addEventListener('keyup', function (e) { // Bind the
keyup event for the grid.
            if ((e.target as any).classList.contains('custemp')) { // Based
on this condition, you can find whether the target is an input element or
not.

                var row = parentsUntil(e.target as any, 'e-row');
                var rowIndex = (row as any).rowIndex; // Get the row index.
                var uid = row.getAttribute('data-uid');
                var rowData = grid.getRowObjectFromUID(uid).data; // Get the
row data.

                (rowData as any).Freight = (e.target as any).value; //
Update the new value for the corresponding column.
                grid.updateRow(rowIndex, rowData); // Update the modified
value in the row data.
            }
        });
    }
});

```

```

    }
  });
  grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <script id="template" type="text/x-template">
    <input id='{$OrderID}' value='{$Freight}' class='custemp'
type='text' style='width: 100%'>
  </script>
  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Customize dialog when editing](#)
- [Cascading DropDownList with Grid Editing](#)
- [Customize the Edit Dialog](#)
- [Tab Inside the Dialog Template](#)
- [How to bulk edit columns in Grid](#)
- [How to use Document Editor as an edit field in Data Grid](#)
- [How to render custom confirmation dialog while updating edit operations](#)
- [How to render ColorPicker component for particular column while editing a record](#)
- [How to positioning the validation error message](#)

Edit types in EJ2 JavaScript Grid control

Customize editors using params

The [columns.editType](#) is used to define the editor component for any particular column. You can set the [columns.editType](#) based on data type of the column.

- [NumericTextBox](#) component for integers, double, and decimal data types.
- [TextBox](#) component for string data type.
- [DropDownList](#) component to show all unique values related to that field.
- [CheckBox](#) component for boolean data type.
- [DatePicker](#) component for date data type.
- [DateTimePicker](#) component for date time data type.

Also, you can customize the behavior of the editor component through the [columns.edit.params](#).

The following table describes editor component and their example edit params of the column.

Component | Example

[NumericTextBox](#) | params: { decimals: 2, value: 5 }

[DropDownList](#) | params: { value: 'Germany' }

[Checkbox](#) | params: { checked: true }

[DatePicker](#) | params: { format: 'dd.MM.yyyy' }

[DateTimePicker](#) | params: { value: new Date() }

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,

```

```

        toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
        editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
            { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
            { field: 'OrderDate', headerText: 'Order Date', type: 'date', width:
120, format: { type: 'date', format: 'dd.MM.yyyy' }, editType:
'datepickeredit', edit: { params: { format: 'dd.MM.yy' } } },
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2', edit: { params: {
decimals: 2, value: 5 } } },
            { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150, edit: { params: { value: 'Germany' } } },
            { field: 'Verified', displayAsCheckBox: true, editType:
'booleanedit', textAlign: 'Center', width: 100, edit: { params: { checked:
true } }
        ],
        height: 265
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If edit type is not defined in the column, then it will be considered as the **stringedit** type (Textbox component).

Restrict to type decimal points in a NumericTextBox while editing the numeric column

By default, the number of decimal places will be restricted to two in the NumericTextBox while editing the numeric column. We can restrict to type the decimal points in a NumericTextBox by using the **validateDecimalOnType** and **decimals** properties of NumericTextBox.

In the below demo, while editing the row we have restricted to type the decimal point value in the NumericTextBox of **Freight** column.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({

```

```

        dataSource: data,
        toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
        editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 150, isPrimaryKey: true },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            {
                field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 150, edit: {
                    params: {
                        validateDecimalOnType: true,
                        decimals: 0,
                        format: "N"
                    }
                }
            },
            { field: 'ShipCity', headerText: 'Ship City', editType:
'dropdownedit', width: 150 }
        ],
        height: 265
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="template" type="text/x-template">
            <a href="#">${ShipCountry}</a>
        </script>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Provide custom data source and enabling filtering to DropDownList

You can provide data source to the DropDownList by using the [columns.edit.params](#) property.

While setting new data source using edit params, you must specify a new **query** property too for the dropdownlist as follows,

```

`ts
{
  params: {
    query: new Query(),
    dataSource: country,
    fields: { value: 'ShipCountry', text: 'ShipCountry' },
    allowFiltering: true
  }
}
`

```

You can also enable filtering for the dropdownlist by passing the **allowFiltering** as **true** to the edit params.

In the below demo, DropDownList is rendered with custom Datasource for the **ShipCountry** column and enabled filtering to search dropdownlist items.

INDEX.TS

```
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { Query } from '@syncfusion/ej2-data';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let country: { [key: string]: Object }[] = [
    { ShipCountry: 'United States', countryId: '1' },
    { ShipCountry: 'Australia', countryId: '2' },
    { ShipCountry: 'India', countryId: '2' }
];
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Normal' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100, isPrimaryKey: true, validationRules: { required: true } },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120, validationRules: { required: true, minLength: 3 } },
        {
            field: 'ShipCountry', headerText: 'Ship Country', width: 120,
            editType: 'dropdownedit', edit: {
                params: {
                    query: new Query(),
                    dataSource: country,
                    fields: { value: 'ShipCountry', text: 'ShipCountry' },
                    allowFiltering: true
                }
            }
        }
    ],
    height: 265
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Open popup while focusing the edit dropdown list

You can open the DropDownList popup while focusing the cell by invoking the [showPopup](#) method inside the ['focus'](#) event of DropDownList component.

In the below demo, we have bound the focus event for the edit DropDownList using the [columns.edit.params](#) property.

INDEX.TS

```
import { Grid, Edit, Toolbar, EJ2Intance } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', edit: { params:
{ focus: ddFocus } }, editType: 'dropdownedit', width: 150 }
  ],
  height: 265
});
grid.appendTo('#Grid');
function ddFocus(e: {event: MouseEvent | KeyboardEvent | TouchEvent}): void
{
  ((e.event.target as HTMLElement).querySelector('.e-dropdownlist') as
EJ2Intance).ej2_instances[0].showPopup();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom editors using template

The cell edit template is used to add a custom component for a particular column by invoking the following functions:

- **create** - It is used to create the element at the time of initialization.
- **write** - It is used to create the custom component or assign default value at the time of editing.
- **read** - It is used to read the value from the component at the time of save.
- **destroy** - It is used to destroy the component.

Render TimePicker component while editing

Use the cell edit template feature of the Grid to render the TimePicker component in the Grid edit form. In the below sample, we have rendered TimePicker component in the **OrderDate** column.

INDEX.TS

```
import { Grid, Edit, Toolbar, Page, Column } from '@syncfusion/ej2-grids';
import { purchaseData } from './datasource.ts';
import { TimePicker } from '@syncfusion/ej2-calendars';
import { enableRipple } from '@syncfusion/ej2-base';
Grid.Inject(Edit, Toolbar, Page);
let ddElem: HTMLElement;
let timeObject: TimePicker;
let grid: Grid = new Grid({
  dataSource: purchaseData,
  allowPaging: true,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true
},
  columns: [
    { field: 'OrderID', headerText: 'Order ID', type: 'number',
isPrimaryKey: true, validationRules: { required: true }, textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', type: 'string', width:
140 },
    { field: 'Freight', headerText: 'Freight', type: 'number', editType:
'numericedit', format: 'C2', textAlign: 'Right', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', type: 'date', format:
'hh:mm', width: 150, edit: {
      create: createOrderDateFn,
      destroy: destroyOrderDateFn,
      read: readOrderDateFn,
      write: writeOrderDateFn }
    }
  ],
  pageSettings: { pageSize: 7 },
  height: 255,
});
grid.appendTo('#Grid');
function createOrderDateFn() {
  ddElem = document.createElement('input');
  return ddElem;
}
function destroyOrderDateFn() {
  timeObject.destroy();
}
```

```
function readOrderDateFn() {
    return timeObject.value;
}
function writeOrderDateFn(args) {
    enableRipple(true);
    timeObject = new TimePicker({
        value: args.rowData[args.column.field],
        step: 60
    });
    timeObject.appendTo(ddElem);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
    </style>
```

```

    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render AutoComplete component while editing

Use the cell edit template feature of the Grid to render the AutoComplete component in the Grid edit form. In the below sample, we have rendered AutoComplete component in the **CustomerID** column.

INDEX.TS

```

import { Grid, Edit, Toolbar, Page, Column } from '@syncfusion/ej2-grids';
import { purchaseData } from './datasource.ts';
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
Grid.Inject(Edit, Toolbar, Page);
let inputEle: HTMLElement;
let autoCompleteIns: AutoComplete;
let autoCompleteData: { [key: string]: Object }[] = [
    { CustomerID: 'VINET', Id: '1' },
    { CustomerID: 'TOMSP', Id: '2' },
    { CustomerID: 'HANAR', Id: '3' },
    { CustomerID: 'VICTE', Id: '4' },
    { CustomerID: 'SUPRD', Id: '5' }
];
let grid: Grid = new Grid({
    dataSource: purchaseData,
    allowPaging: true,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true },
    columns: [

```

```

    { field: 'OrderID', headerText: 'Order ID', type: 'number',
    isPrimaryKey: true, validationRules: { required: true }, textAlign: 'Right',
    width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', type: 'string', width:
    140, edit: {
        create: createCustomerIDFn,
        destroy: destroyCustomerIDFn,
        read: readCustomerIDFn,
        write: writeCustomerIDFn }
    },
    { field: 'Freight', headerText: 'Freight', type: 'number', editType:
    'numericedit', format: 'C2', textAlign: 'Right', width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', type: 'date', format:
    'yMd', editType: 'datepickeredit', width: 150 }
    ],
    pageSettings: { pageSize: 7 },
    height: 255,
});
grid.appendTo('#Grid');
function createCustomerIDFn() {
    inpuEle = document.createElement('input');
    return inpuEle;
}
function destroyCustomerIDFn() {
    autoCompleteIns.destroy();
}
function readCustomerIDFn() {
    return autoCompleteIns.value;
}
function writeCustomerIDFn(args) {
    autoCompleteIns = new AutoComplete({
        allowCustom: true,
        value: args.rowData[args.column.field],
        dataSource: autoCompleteData,
        fields: { value: 'CustomerID', text: 'CustomerID' }
    });
    autoCompleteIns.appendTo(inpuEle);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    popups/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render MultiSelect DropDown component while editing

Use the cell edit template feature of the Grid to render the MultiSelect DropDown component in the Grid edit form. In the below sample, we have rendered MultiSelect DropDown component in the **ShipCity** column.

INDEX.TS

```
import { Grid, Edit, Toolbar, Page, Column } from '@syncfusion/ej2-grids';
import { purchaseData } from './datasource.ts';
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
Grid.Inject(Edit, Toolbar, Page);
let ddElem: HTMLElement;
let multiSelectObj: MultiSelect;
let multiselectDatasource = [
    { ShipCity: 'Reims', Id: '1' },
    { ShipCity: 'Münster', Id: '2' },
    { ShipCity: 'Rio de Janeiro', Id: '3' },
    { ShipCity: 'Lyon', Id: '4' },
    { ShipCity: 'Charleroi', Id: '5' }
];
let grid: Grid = new Grid({
    dataSource: purchaseData,
    allowPaging: true,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', type: 'number',
isPrimaryKey: true, validationRules: { required: true }, textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', type: 'string', width:
140 },
        { field: 'Freight', headerText: 'Freight', type: 'number', editType:
'numericedit', format: 'C2', textAlign: 'Right', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', type: 'string', width:
180, edit: {
            create: createShipCityFn,
            read: readShipCityFn,
            destroy: destroyShipCityFn,
            write: writeShipCityFn }
        }
    ],
    pageSettings: { pageSize: 7 },
    height: 255,
});
grid.appendTo('#Grid');
function createShipCityFn() {
    ddElem = document.createElement('input');
    return ddElem;
}
function readShipCityFn() {
    return multiSelectObj.value.join(', ');
}
function destroyShipCityFn() {
    multiSelectObj.destroy();
}
function writeShipCityFn(args) {
```

```

{
    let multiSelectVal = args.rowData[args.column.field]
        ? args.rowData[args.column.field].split(',')
        : [];
    multiSelectObj = new MultiSelect({
        value: multiSelectVal,
        dataSource: multiSelectDatasource,
        fields: { value: 'ShipCity', text: 'ShipCity' },
        floatLabelType: 'Never',
        mode: 'Box'
    });
    multiSelectObj.appendTo(ddElem);
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
    </style>

```

```

    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render MaskedTextBox component while editing

Use the cell edit template feature of the Grid to render the MaskedTextBox component in the Grid edit form. In the following sample, the MaskedTextBox component is rendered in the Mask column.

INDEX.TS

```

import { Grid, Edit, Toolbar, Column } from '@syncfusion/ej2-grids';
import { MaskedTextBox } from '@syncfusion/ej2-inputs';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let elem: HTMLElement;
let maskObj: MaskedTextBox;
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        {
            field: 'Mask', headerText: 'Mask', width: 140, edit: {

```

```

        create: () => {
            elem = document.createElement('input');
            return elem;
        },
        read: () => {
            return maskObj.value;
        },
        destroy: () => {
            maskObj.destroy();
        },
        write: (args: { rowData: Object, column: Column }) => {
            maskObj = new MaskedTextBox({
                mask: "0-0-0-0",
                value: args.rowData.Mask
            });
            maskObj.appendTo((args as any).element);
        }
    }
    ],
    height: 265
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render RichTextEditor component while editing

Use the cell edit template feature of the Grid to render the RichTextEditor component in the Grid edit form. In the below sample, we have rendered RichTextEditor component in the **ShipAddress** column, so we use [allowTextWrap](#) property to true.

INDEX.TS

```

import { Grid, Edit, Page, Column, Toolbar } from '@syncfusion/ej2-grids';
import { RichTextEditor, Link, Toolbar as RTEToolbar, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-richtexteditor';
import { purchaseData } from './datasource.ts';
RichTextEditor.Inject(Link, RTEToolbar, Image, HtmlEditor, QuickToolbar);
Grid.Inject(Edit, Toolbar, Page);
let rteElement: HTMLElement;
let richtextEditor: RichTextEditor;

```

```

let grid: Grid = new Grid({
  dataSource: purchaseData,
  allowPaging: true,
  allowTextWrap: true,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true
},
  created: function (args) {
    this.keyConfigs.enter = '';
  },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', type: 'number',
isPrimaryKey: true, validationRules: { required: true }, textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', type: 'string', width:
140 },
    { field: 'Freight', headerText: 'Freight', type: 'number', editType:
'numericedit', format: 'C2', textAlign: 'Right', width: 120 },
    { field: 'ShipAddress', headerText: 'Ship Address', type: 'string',
valueAccessor: valueAccessor, disableHtmlEncode: false, width: 180, edit: {
      create: createShipAddressFn,
      read: readShipAddressFn,
      write: writeShipAddressFn,
      destroy: destroyShipAddressFn }
    }
  ],
  pageSettings: { pageSize: 7 },
  height: 255,
});
grid.appendTo('#Grid');
function createShipAddressFn() {
  rteElement = document.createElement('textarea');
  return rteElement;
}
function readShipAddressFn() {
  return richtextEditor.value;
}
function writeShipAddressFn(args) {
  richtextEditor = new RichTextEditor({
    value: args.rowData[args.column.field],
  });
  richtextEditor.appendTo(rteElement);
}
function destroyShipAddressFn() {
  richtextEditor.destroy();
}
function valueAccessor(field, data, column) {
  var value = data[field];
  if (value != undefined) {
    return value.split('\n').join('<br>');
  } else {
    return '';
  }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

    <div id="container">
      <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render multiple columns in DropDownList component while editing

Use the cell edit template feature of the Grid to render the DropDownList component in the Grid edit form.

The DropDownList has been provided with several options to customize each list item, group title, selected value, header, and footer element. By default, list items can be rendered as a single column in the DropDownList component. Instead of this, multiple columns can be rendered. This can be achieved by using the [headerTemplate](#) and [itemTemplate](#) properties of the DropDownList component.

This is demonstrated in the following sample.

INDEX.TS

```

import { Grid, Edit, Toolbar, Column } from '@syncfusion/ej2-grids';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Query } from '@syncfusion/ej2-data';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let element: HTMLElement;
let dropdownobj: DropDownList;
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
    { field: 'ShipCountry', width: 300, headerText: 'ShipCountry' ,
edit: {
      create: function () {
        element = document.createElement('input');
        return element;
      },
      read: function () {
        return dropdownobj.value;
      },
      destroy: function () {
        dropdownobj.destroy();
      },
      write: function (args) {

```

```

        dropdownobj = new DropDownList({
            dataSource:data,
            value: args.rowData[args.column.field],
            query: new Query().select(['EmployeeID',
'ShipCountry']).take(10),
            fields: { text: 'ShipCountry', value: 'ShipCountry' },
            placeholder: 'Select a Country',
            headerTemplate:
'<table><tr><th>EmployeeID</th><th>ShipCountry</th></tr></table>',
            itemTemplate: '<div class="e-grid"><table class="e-
table"><tbody><tr><td class="e-rowcell">${EmployeeID}</td><td class="e-
rowcell">${ShipCountry}</td></tr> </tbody></table></div>'
        });
        dropdownobj.appendTo(element);
    }
}
}
]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .content {

```

```

        margin: 0 auto;
        width: 550px;
    }
    table{
        width:100%;
        border-collapse: separate;
        table-layout: fixed;
    }
    th,td{
        border-width: 1px 0 0 1px;
        border-color: #e0e0e0;
        text-align: left;
        border-style: solid;
        display: table-cell;
    }
    th{
        line-height: 36px;
        text-indent: 16px;
    }
    .e-ddl-header{
        padding-right: 17px;
        border-width: 1px 0px 1px 0px;
        border-color: #e0e0e0;
        border-style: solid;
    }
    .e-dropdownbase .e-list-item{
        padding-right:0px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Access editor components

You can access the component instance from the component element using the `ej2_instances` property.

In the below demo, you can access the Editor component instance while adding or editing actions on the [actionComplete](#) event.

INDEX.TS

```
import { Grid, Edit, Toolbar, EditEventArgs, EJ2Intance } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    actionComplete: (args: EditEventArgs) => {
        if (args.requestType === 'beginEdit' || args.requestType === 'add')
        {
            let tr: Element = args.row;
            let numericTextBox: Element = tr.querySelector('.e-
numerictextbox'); // numeric TextBox component element
            if (numericTextBox) {
                console.log('NumericTextBox instance: ',
(<EJ2Intance>numericTextBox).ej2_instances[0]); // numeric TextBox instance
            }
            let dropDownList: Element = tr.querySelector('.e-dropdownlist');
            // dropDownList component element
            if (dropDownList) {
                console.log('DropDownList instance: ',
(<EJ2Intance>dropDownList).ej2_instances[0]); // dropDownList instance
            }
        }
    },
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 265
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

In line editing in EJ2 JavaScript Grid control

In Normal edit mode, when you start editing the currently selected record is changed to edit state. You can change the cell values and save edited data to the data source. To enable Normal edit, set the [editSettings.mode](#) as **Normal**.

INDEX.TS

```
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
  ],
  height: 265
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Normal edit mode is default mode of editing.

Automatically update the column based on another column edited value

You can update the column value based on another column edited value by using the Cell Edit Template feature.

In the below demo, we have update the **TotalCost** column value based on the **UnitPrice** and **UnitInStock** column value while editing.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { productData } from './productData.ts';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
Grid.Inject(Edit, Toolbar);
var priceElem: HTMLElement;;
var priceObj: NumericTextBox;
var stockElem: HTMLElement;;
var stockObj: NumericTextBox;
let grid: Grid = new Grid({
  dataSource: productData,
  editSettings: {
    allowEditing: true,
    allowAdding: true,
    allowDeleting: true,
    mode: 'Normal',
    newRowPosition: 'Top'
  },
  allowPaging: true,
  pageSettings: { pageCount: 5 },
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  columns: [
    {
      field: 'ProductID',
      isPrimaryKey: true,
      headerText: 'Product ID',
      textAlign: 'Right',
      validationRules: { required: true, number: true },
      width: 140
    },
    {
      field: 'ProductName',
      headerText: 'Product Name',
      validationRules: { required: true },
      width: 140
    },
    {
      field: 'UnitPrice',
      headerText: 'UnitPrice',
      textAlign: 'Right',
      edit: {
        create: function() {
          priceElem = document.createElement('input');
          return priceElem;
        },
        read: function() {
          return priceObj.value;
        },
        destroy: function() {
          priceObj.destroy();
        },
        write: function(args) {
          priceObj = new NumericTextBox({
            value: args.rowData[args.column.field],
            change: function(args) {

```



```

        var formEle =
grid.element.querySelector('form').ej2_instances[0];
        var totalCostFieldEle = formEle.getInputElement('TotalCost');
        totalCostFieldEle.value = priceObj.value * stockObj.value;
    }
    });
    priceObj.appendTo(priceElem);
}
},
width: 140,
format: 'C2',
validationRules: { required: true }
},
{
    field: 'UnitsInStock',
    headerText: 'Units In Stock',
    textAlign: 'Right',
    edit: {
        create: function() {
            stockElem = document.createElement('input');
            return stockElem;
        },
        read: function() {
            return stockObj.value;
        },
        destroy: function() {
            stockObj.destroy();
        },
        write: function(args) {
            stockObj = new NumericTextBox({
                value: args.rowData[args.column.field],
                change: function(args) {
                    var formEle =
grid.element.querySelector('form').ej2_instances[0];
                    var totalCostFieldEle = formEle.getInputElement('TotalCost');
                    totalCostFieldEle.value = priceObj.value * stockObj.value;
                }
            });
            stockObj.appendTo(stockElem);
        }
    },
    width: 140,
    validationRules: { required: true }
},
{
    field: 'TotalCost',
    headerText: 'Total Unit Cost',
    textAlign: 'Right',
    allowEditing: false,
    width: 140,
    format: 'C2',
}
]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cancel edit based on condition

You can prevent the CRUD operations of the Grid by using condition in the [actionBegin](#) event with requestType as `beginEdit` for editing, `add` for adding and `delete` for deleting actions.

In the below demo, we prevent the CRUD operation based on the `Role` column value. If the Role Column is `Employee`, we are unable to edit/delete that row.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let isAddable: boolean = true;
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'Role', headerText: 'Role', width: 120, },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    actionBegin: actionBegin,
    height: 240
});
grid.appendTo('#Grid');
function actionBegin(args) {
    if (args.requestType == 'beginEdit') {
        if (args.rowData['Role'].toLowerCase() == 'employee') {
            args.cancel = true;
        }
    }
    if (args.requestType == 'delete') {
        if (args.data[0]['Role'].toLowerCase() == 'employee') {
            args.cancel = true;
        }
    }
}

```

```

    }
    if (args.requestType == 'add') {
        if (!isAddable) {
            args.cancel = true;
        }
    }
}
}
var button = document.createElement('button');
button.innerText = 'Grid is Addable';
document.body.insertBefore(button, document.body.children[0])
button.addEventListener('click', btnClick.bind(this));
function btnClick(args) {
    args.target.innerText == 'Grid is Addable' ? (args.target.innerText =
'Grid is Not Addable') : (args.target.innerText = 'Grid is Addable');
    isAddable = !isAddable;
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
    </style>

```

```

        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Perform CRUD action programmatically

Grid methods can be used to perform CRUD operations programmatically. The [addRecord](#), [deleteRecord](#), and [startEdit](#) methods are used to perform CRUD operations in the following demo.

- To add a new record to the Grid, use the [addRecord](#) method. In this method, you can pass the data parameter to add a new record to the Grid, and the index parameter to add a record at a specific index. If you call this method with no parameters, it will create an empty row in the Grid.
- To change the selected row to the edit state, use the [startEdit](#) method.
- If you need to update the row data in the Grid's datasource, you can use the [updateRow](#) method. In this method, you need to pass the index value of the row to be updated along with the updated data.
- If you need to update the particular cell in the row, you can use the [setCellValue](#) method. In this method, you need to pass the primary key value of the data source, field name, and new value for the particular cell.
- To remove a selected row from the Grid, use the [deleteRecord](#) method. For both edit and delete operations, you must select a row first.

Note: In both normal and dialog editing modes, these methods can be used.

INDEX.TS

```
import { Grid, Edit } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit);
let grid: Grid = new Grid({
    dataSource: data,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 230
});
grid.appendTo('#Grid');
document.getElementById('add').onclick = () => {
    grid.addRecord({ "OrderID": "10248", "CustomerID": "RTER", "ShipCity":
"America", "ShipName": "Hanari" });
};
document.getElementById('edit').onclick = () => {
    grid.startEdit();
};
document.getElementById('delete').onclick = () => {
    grid.deleteRecord();
};
document.getElementById('updaterow').onclick = () => {
    grid.updateRow(0, { OrderID: 10248, CustomerID: 'RTER', ShipCity:
'America', ShipName: 'Hanari' });
};
document.getElementById('updatecell').onclick = () => {
    grid.setCellValue((grid.currentViewData[0] as
any).OrderID, 'CustomerID', 'Value Changed');
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="edit">Edit</button>
        <button id="add">Add</button>
        <button id="delete">Delete</button>
        <button id="updaterow">Update Row</button>
        <button id="updatecell">UpdateCell</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Confirmation dialog

The delete confirm dialog can be shown when deleting a record by defining the [showDeleteConfirmDialog](#) as `true`

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],

```

```

        editSettings: { showDeleteConfirmDialog: true, allowEditing: true,
allowAdding: true, allowDeleting: true, mode: 'Normal' },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
            { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
            { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
        ],
        height: 265
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
    </style>

```



```

    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The `showDeleteConfirmDialog` supports all type of edit modes.

Default column values on add new row

The grid provides an option to set the default value for the columns when adding a new record in it. To set a default value for the particular column by defining the [columns.defaultValue](#).

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true } },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true, minLength: 3 }, defaultValue: 'HANAR' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ]
});

```

```

    ],
    height: 265
  });
  grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>

```

```

</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding a new row at the bottom of the Grid

By default, a new row will be added at the top of the grid. You can change it by setting [editSettings.newRowPosition](#) as **Bottom**.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, newRowPosition: 'Bottom' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 265
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">

```

```

    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
    <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add `newRowPosition` is supported for **Normal** and **Batch** editing modes.

Move the focus to a particular cell instead of first cell while editing a row

The [recordDoubleClick](#) event allows you to move the focus to the corresponding cell (the cell that you doubled-clicked to edit a row) instead of the first cell in edit form. With the help of this event, you can focus the double-clicked column in inline edit mode.

INDEX.TS

```

import { Grid, Page, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, Edit);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true,
    mode: "Normal"
    },
    recordDoubleClick: recordDoubleClick,
    actionComplete: actionComplete,
    columns: [
        { field: 'OrderID', isPrimaryKey: true, headerText: 'Order ID',
textAlign: 'Right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', editType: "numericedit",
textAlign: 'Right', width: 120, format: 'C2' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, editType: "datetimepickeredit",
format: { type: "dateTime", format: "M/d/y hh:mm a" }, },
        { field: "ShipCountry", headerText: "Ship Country", editType:
"dropdownedit", width: 150, edit: { params: { popupHeight: "300px" } }
        }
    ],
    height: 220
});
grid.appendTo('#Grid');
var fieldName;
function recordDoubleClick(e) {
    var clickedColumnIndex = e.cell.getAttribute("data-colindex");
    fieldName = this.columnModel[parseInt(clickedColumnIndex)].field;
}
function actionComplete(e) {
    if (e.requestType === "beginEdit") {
        // focus the column
        e.form.elements[grid.element.getAttribute("id") + fieldName].focus();
    }
}

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Dialog editing in EJ2 JavaScript Grid control

In Dialog edit mode, when you start editing the currently selected row data will be shown on a dialog. You can change the cell values and save edited data to the data source. To enable Dialog edit, set the [editSettings.mode](#) as **Dialog**.

INDEX.TS

```
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
  ],
  height: 265
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize edit dialog

You can customize the Header and Footer appearance of the edit dialog in the [actionComplete](#) event based on [requestType](#) as [beginEdit](#) or [add](#).

In the following example, We have customized the dialog's height and title of the dialog's header and also we have changed the button content name in the dialog's footer.

INDEX.TS


```

import { Grid, Edit, Toolbar, DialogEditEventArgs } from '@syncfusion/ej2-
grids';
import { data } from './datasource.ts';
import { Dialog } from '@syncfusion/ej2-popups';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog' },
    actionComplete: (args: DialogEditEventArgs) => {
        if ((args.requestType === 'beginEdit' || args.requestType
=== 'add')) {
            let dialog: Dialog = args.dialog;
            // set the height of the dialog
            dialog.height = 400;
            // change the header of the dialog
            dialog.header = args.requestType === 'beginEdit' ? 'Edit
Record of ' + args.rowData['CustomerID'] : 'New Customer';
            // change the footer button name of the dialog
            dialog.getButtons()[0].content = "Ok";
            dialog.getButtons()[1].content = "Close";
        }
    },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right',
            width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width:
120, },
        { field: 'ShipCountry', headerText: 'Ship Country', width:
150 }],
    height: 265
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The Grid add or edit dialog element has the max-height property, which is calculated based on the available window height. So, in the normal window (1920 x 1080), it is possible to set the dialog's height up to 658px.

Show or hide columns in dialog editing

The Grid has the option to show hidden columns or hide visible columns while editing in the dialog edit mode by using the [actionBegin](#) event of the Grid.

In the `actionBegin` event, when the `requestType` is `beginEdit` or `add`, the column will be shown or hidden using the `column.visible` property. When the `requestType` is `save`, the properties will be reset to their original state.

In the following example, the CustomerID column is rendered as a hidden column, and the ShipCountry column is rendered as a visible column. In the edit mode, the CustomerID column will be changed to a visible state and the ShipCountry column will be changed to a hidden state.

INDEX.TS

```
import { Grid, Edit, Toolbar, DialogEditEventArgs } from '@syncfusion/ej2-
grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120,
visible: false },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 265,
    actionBegin: function(args: DialogEditEventArgs) {
        if ((args.requestType === 'beginEdit' || args.requestType ===
'add')) {
            for (var i = 0; i < this.columns.length; i++) {
                if (this.columns[i].field == "CustomerID") {
                    this.columns[i].visible = true;
                }
                else if (this.columns[i].field == "ShipCountry") {
                    this.columns[i].visible = false;
                }
            }
        }
        if (args.requestType === 'save') {
            for (var i = 0; i < this.columns.length; i++) {
                if (this.columns[i].field == "CustomerID") {
                    this.columns[i].visible = false;
                }
                else if (this.columns[i].field == "ShipCountry") {
                    this.columns[i].visible = true;
                }
            }
        }
    }
});
```

```

    }
  }
}
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;

```

```

        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Use wizard like dialog editing

Wizard helps you to create intuitive step-by-step forms to fill. You can achieve the wizard-like editing by using the dialog template feature. It supports your own editing template by defining the [editSettings.mode](#) as `Dialog` and [editSetting.template](#) as SCRIPT element ID or HTML string which holds the template.

The following example demonstrates the wizard-like editing in the grid with obtrusive validation.

INDEX.TS

```

import { Grid, Edit, Toolbar, DialogEditEventArgs } from '@syncfusion/ej2-
grids';
import { data } from './datasource.ts';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { DataUtil } from '@syncfusion/ej2-data';
import { CheckBox } from '@syncfusion/ej2-buttons';
Grid.Inject(Edit, Toolbar);
let countryData: {}[] = DataUtil.distinct(data, 'ShipCountry', true);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog', template: '#dialogtemplate' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true }},
        { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true }},
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 265,
    actionComplete: (args: DialogEditEventArgs) => {

```

```

        if ((args.requestType === 'beginEdit' || args.requestType ===
        'add')) {
            new DropDownList({value: args.rowData.ShipCountry, popupHeight:
            '300px', floatLabelType: 'Always',
                dataSource: countryData, fields: {text: 'ShipCountry',
            value: 'ShipCountry'}, placeholder: 'Ship Country'},
            args.form.elements.namedItem('ShipCountry') as HTMLInputElement);
            new CheckBox({ label: 'Verified', checked: args.rowData.Verified
            }, args.form.elements.namedItem('Verified'));
            // Set initail Focus
            if (args.requestType === 'beginEdit') {
                (args.form.elements.namedItem('CustomerID') as
            HTMLInputElement).focus();
            }
            initializeWizard();
        }
    }
});
grid.appendTo('#Grid');
function initializeWizard() {
    let currentTab = 0;
    document.getElementById('nextBtn').onclick = function () {
        if (validate()) {
            if (this.innerHTML !== 'SUBMIT'){
                currentTab++;
                nextpre(currentTab);
            } else {
                grid.endEdit();
            }
        }
    }
    function validate(tab) {
        let valid: boolean = true;
        [].slice.call(document.getElementById('tab' +
currentTab).querySelectorAll('[name]')).forEach(element => {
            element.form.ej2_instances[0].validate(element.name);
            if (element.getAttribute('aria-invalid') === 'true'){
                valid = false;
            }
        });
        if (!valid) {
            return false;
        }
        return true;
    }
    document.getElementById('prevBtn').onclick = function () {
        if (validate()) {
            currentTab--;
            nextpre(currentTab);
        }
    }
}
function nextpre(current) {
    let tabs: HTMLInputElement[] =
[...].slice.call(document.getElementsByClassName('tab'))
    tabs.forEach(element => element.style.display = 'none');
    tabs[current].style.display = '';
}

```

```

    if(current) {
        document.getElementById('prevBtn').style.display = '';
        document.getElementById('nextBtn').innerHTML = 'SUBMIT';
    } else {
        document.getElementById('prevBtn').style.display = 'none';
        document.getElementById('nextBtn').innerHTML = 'NEXT';
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/css/bootstrap.min.c
ss">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="Grid"></div>
</div>
<script id="dialogtemplate" type="text/x-template">
  <div>
    <div id="tab0" class='tab'>
      <div class="form-row">
        <div class="form-group col-md-6">
          <div class="e-float-input e-control-wrapper">
            <input id="OrderID" required name="OrderID"
type="text" value=${if(isAdd)} '' ${else} ${OrderID} ${if} ${if(isAdd)} ''
${else} disabled ${if} />
            <span class="e-float-line"></span>
            <label class="e-float-text e-label-top"
for="OrderID">Order ID</label>
          </div>
        </div>
      </div>
      <div class="form-row">
        <div class="form-group col-md-6">
          <div class="e-float-input e-control-wrapper">
            <input id="CustomerID" required
name="CustomerID" type="text" value=${if(isAdd)} '' ${else} ${CustomerID}
${if} />
            <span class="e-float-line"></span>
            <label class="e-float-text e-label-top"
for="CustomerID">Customer ID</label>
          </div>
        </div>
      </div>
    </div>
    <div id=tab1 style="display: none" class='tab'>
      <div class="form-row">
        <div class="form-group col-md-6">
          <input type="text" name="ShipCountry"
id="ShipCountry" value=${if(isAdd)} '' ${else} ${ShipCountry} ${if} />
        </div>
      </div>
      <div class="form-row">
        <div class="form-group col-md-6">
          <input type="checkbox" name="Verified" id="Verified"
${if(Verified)} checked ${if} />
        </div>
      </div>
    </div>
    <div id='footer'>
      <button id="prevBtn" class="e-info e-btn" type="button"
style="display: none; float: left">Previous</button>

      <button id="nextBtn" class="e-info e-btn" type="button"
style="float: right">Next</button>
    </div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {

```



```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize Add/Edit Dialog footer

In dialog edit mode, a dialog will show up when editing the currently selected row or adding a new row. By default, you can save or cancel the edited changes by clicking the Save or Cancel button in the dialog's footer. Along with these buttons, it is possible to add a custom button in the footer section using the [actionComplete](#) event of the Grid.

In the following sample, using the `dialog` argument of the `actionComplete` event, the action for the custom button can be customized.

INDEX.TS

```

import { Grid, Edit, Toolbar, DialogEditEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete'],
  editSettings: {
    allowEditing: true,
    allowAdding: true,
    allowDeleting: true,
    mode: 'Dialog',
  },
  actionComplete: actionComplete,
  columns: [{
    field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', editType: 'dropdownedit', width: 150 },
  ]
});
grid.appendTo('#Grid');
function actionComplete(args) {
  if (args.requestType === 'beginEdit' || args.requestType === 'add') {
    let newFooterButton = {
      buttonModel: { content: 'custom' },
      click: onCustomButtonClick
    };
    args.dialog.buttons.push(newFooterButton);
    args.dialog.refresh();
  }
}
function onCustomButtonClick() {
  alert('Add/Edit dialog custom footer button clicked');
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Template editing in EJ2 JavaScript Grid control

Inline or dialog template editing

The dialog/inline template editing provides an option to customize the default behavior of dialog editing. Using the dialog template, you can render your own editors by defining the [editSettings.mode](#) as Dialog/Inline and [editSetting.template](#) as SCRIPT element ID or HTML string which holds the template.

In some cases, you need to add the new field editors in the dialog which are not present in the column model. In that situation, the dialog template will help you to customize the default edit dialog.

In the following sample, grid enabled with dialog template editing.

INDEX.TS

```

import { Grid, Edit, Toolbar, DialogEditEventArgs } from '@syncfusion/ej2-
grids';
import { data } from './datasource.ts';
import { DataUtil } from '@syncfusion/ej2-data';
import { DataUtil } from '@syncfusion/ej2-data';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { CheckBox } from '@syncfusion/ej2-buttons';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
Grid.Inject(Edit, Toolbar);
let countryData: {}[] = DataUtil.distinct(data, 'ShipCountry', true);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog', template: '#dialogtemplate' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 265,
    actionBegin: (args: SaveEventArgs) => {
        if (args.requestType === 'save') {

```

```

        // cast string to integer value.
        args.data['Freight'] =
parseFloat(args.form.querySelector("#Freight").value);
    }
},
    actionComplete: (args: DialogEditEventArgs) => {
        if ((args.requestType === 'beginEdit' || args.requestType ===
'add')) {
            // Add Validation Rules
            args.form.ej2_instances[0].addRules('Freight', {max: 500});
            // EJ2-control Rendering
            new DropDownList({value: args.rowData.ShipCountry, popupHeight:
'200px', floatLabelType: 'Always',
                dataSource: countryData, fields: {text: 'ShipCountry',
value: 'ShipCountry'}, placeholder: 'Ship Country'},
args.form.elements.namedItem('ShipCountry') as HTMLInputElement);
            new CheckBox({ label: 'Verified', checked: args.rowData.Verified
}, args.form.elements.namedItem('Verified'));
            new NumericTextBox({value: args.rowData.Freight, format: 'C2',
placeholder: 'Freight', floatLabelType: 'Always' },
args.form.elements.namedItem('Freight') as HTMLInputElement );
            // Set initail Focus
            if (args.requestType === 'beginEdit') {
                (args.form.elements.namedItem('CustomerID') as
HTMLInputElement).focus();
            }
        }
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/css/bootstrap.min.c
ss">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
    <script id="dialogtemplate" type="text/x-template">
        <div>
            <div class="form-row" >
                <div class="form-group col-md-6">
                    <div class="e-float-input e-control-wrapper">
                        <input id="OrderID" name="OrderID" type="text"
value=${if(isAdd)} '' ${else} ${OrderID} ${if} ${if(isAdd)} '' ${else}
disabled ${if}/>
                        <span class="e-float-line"></span>
                        <label class="e-float-text e-label-top"
for="OrderID">Order ID</label>
                    </div>
                </div>
                <div class="form-group col-md-6">
                    <div class="e-float-input e-control-wrapper">
                        <input id="CustomerID" name="CustomerID" type="text"
value=${if(isAdd)} '' ${else} ${CustomerID} ${if} />
                        <span class="e-float-line"></span>
                        <label class="e-float-text e-label-top"
for="CustomerID">Customer ID</label>
                    </div>
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <div>
                        <input id="ShipCountry" name="ShipCountry"
type="text" value=${if(isAdd)} '' ${else} ${ShipCountry} ${if} />
                    </div>
                </div>
                <div class="form-group col-md-6">

```

```

        <input id="Freight" value=${if(isAdd)} '' ${else}
${Freight} ${/if} />
    </div>
</div>
<div class="form-row">
    <div class="form-group col-md-12">
        <div class="e-float-input e-control-wrapper">
            <textarea type="text" name="ShipAddress"
id="ShipAddress" value=${if(isAdd)} '' ${else} ${ShipAddress}
${/if}>${if(isAdd)} ${else} ${ShipAddress} ${/if}</textarea>
            <span class="e-float-line"></span>
            <label class="e-float-text e-label-top"
for="ShipAddress">Ship Address</label>
        </div>
    </div>
</div>
<div class="form-row">
    <div class="form-group col-md-6">
        <input type="checkbox" name="Verified" id="Verified"
${if(isAdd)} '' ${else} checked ${/if} />
    </div>
</div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The template form editors should have **name** attribute.

Template context

The Essential JS2 packages has a built-in template engine. Using the template engine, you can access the row information inside the HTML element and bind the attributes, values, or elements based on this row information.

The following properties will be available at the time of template execution.

Property Name	Usage
---------------	-------

isAdd	A Boolean property; it defines whether the current row should be a new record or not.
-------	---

In the following code example, the **OrderID** textbox has been disabled by using the **isAdd** property.

,

// The disabled attributes will be added based on the isAdd property.

```

<input id="OrderID" name="OrderID" type="text" value=${if(isAdd)} '' ${else} ${OrderID} ${/if}
${if(isAdd)} '' ${else} disabled ${/if}/>

```

The following code example illustrates rendering the `OrderID` textbox, when a new record is added.

```

{if(isAdd)}
<div class="form-group col-md-6">
<div class="e-float-input e-control-wrapper">
<input id="OrderID" name="OrderID" type="text" value={if(isAdd)} '' {else} ${OrderID} ${if} />
<span class="e-float-line"></span>
<label class="e-float-text e-label-top" for="OrderID">Order ID</label>
</div>
</div>
{/if}

```

The dialog/inline dialog template syntax supports the ES6 expression string literals, and you can refer to the [Template Engine](#) for more template syntax.

Render editors as components

You can convert the form editors to EJ2 controls in the [actionComplete](#) event based on the `requestType` as `beginEdit` or `add`.

The following code example illustrates rendering the drop-down list control in the `actionComplete` event.

```

`ts
actionComplete: (args: DialogEditEventArgs) => {
if ((args.requestType === 'beginEdit' || args.requestType === 'add')) {
let countryData: {}[] = DataUtil.distinct(data, 'ShipCountry', true);
new DropDownList({value: args.rowData.ShipCountry, popupHeight: '200px', floatLabelType: 'Always',
dataSource: countryData, fields: {text: 'ShipCountry', value: 'ShipCountry'}, placeholder: 'Ship Country'},
args.form.elements.namedItem('ShipCountry') as HTMLInputElement);
}
}

```

Get value from editor

You can read, format, and update the current editor value in the [actionBegin](#) event at the time of setting `requestType` to `save`.

In the following code example, the `freight` value has been formatted and updated.

```

`ts

```

```

actionBegin: (args: SaveEventArgs) => {
  if (args.requestType === 'save') {
    // cast string to integer value.
    args.data['Freight'] = parseFloat(args.form.querySelector("#Freight").value);
  }
}

```

Set focus to editor

By default, the first input element in the dialog will be focused while opening the dialog. If the first input element is in disabled or hidden state, focus the valid input element in the [actionComplete](#) event based on `requestType` as `beginEdit`.

```

`ts
actionComplete: (args: DialogEditEventArgs) => {
  // Set initail Focus
  if (args.requestType === 'beginEdit') {
    (args.form.elements.namedItem('CustomerID') as HTMLInputElement).focus();
  }
}

```

Adding validation rules for custom editors

If you have used additional fields that are not present in the column model, then add the validation rules to the [actionComplete](#) event.

```

`ts
actionComplete: (args: DialogEditEventArgs) => {
  if ((args.requestType === 'beginEdit' || args.requestType === 'add')) {
    // Add Validation Rules
    args.form.ej2_instances[0].addRules('Freight', {max: 500});
  }
}

```

Render tab component inside the dialog template

You can use the [tab](#) component inside the dialog edit UI using the dialog template feature. The dialog template feature can be enabled by defining the [editSettings.mode](#) as `Dialog` and [editSetting.template](#) as SCRIPT element ID or HTML string which holds the template.

To include the tab components in the dialog, follow the given steps:

Step 1:

Initialize the template for your tab component.

```

<div>
<div id="edittab"></div>
<div id="tab1">
<div class="form-row" >
<div class="form-group col-md-6">
<div class="e-float-input e-control-wrapper">
<input id="OrderID" name="OrderID" type="text" value=${if(isAdd)} " ${else} ${OrderID} ${if}
${if(isAdd)} " ${else} disabled ${if} />
<span class="e-float-line"></span>
<label class="e-float-text e-label-top" for="OrderID">Order ID</label>
</div>
</div>
</div>
<div class="form-row" >
<div class="form-group col-md-6">
<div class="e-float-input e-control-wrapper">
<input id="CustomerID" name="CustomerID" type="text" value=${if(isAdd)} " ${else} ${CustomerID}
${if} />
<span class="e-float-line"></span>
<label class="e-float-text e-label-top" for="CustomerID">Customer ID</label>
</div>
</div>
</div>
<button id='next' class='e-info e-btn' style="float: right" type='button'> next</button>
</div>
<div id="tab2" style='display: none'>
<div class="form-row" >
<div class="form-group col-md-6">
<input type="text" name="ShipCountry" id="ShipCountry" value=${if(isAdd)} " ${else} ${ShipCountry}
${if} />
</div>
</div>

```

```

<div class="form-row">
<div class="form-group col-md-6">
<input type="checkbox" name="Verified" id="Verified" ${if(isAdd)} " ${else} checked ${/if} />
</div>
</div>
<button id='submit' class='e-info e-btn' style="float: right" type='button'> submit</button>
</div>
</div>
`

```

Step 2:

To render the tab component, use the [actionComplete](#) event of the grid.

```

`ts
let tabObj: Tab = new Tab({
showCloseButton: false,
selecting: (e) => {if(e.isSwiped) {e.cancel = true;} if(e.selectingIndex === 1) {e.cancel = !validate(1)}},
items: [
{ header: { 'text': 'Details' }, content: '#tab1' },
{ header: { 'text': 'Verify' }, content: '#tab2' },
]
});
tabObj.appendTo('#edittab');
`

```

The following example demonstrates rendering the tab control inside the edit dialog. The tab control contains two tabs. You can fill the first tab and then navigate to the second tab. The validation for first tab will be done before navigating to the second tab.

INDEX.TS

```

import { Grid, Edit, Toolbar, DialogEditEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { DataUtil } from '@syncfusion/ej2-data';
import { Tab } from '@syncfusion/ej2-navigations';
import { CheckBox } from '@syncfusion/ej2-buttons';
Grid.Inject(Edit, Toolbar);
let countryData: {}[] = DataUtil.distinct(data, 'ShipCountry', true);
let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete'],

```

```

editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog', template: '#dialogtemplate' },
columns: [
  { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true } },
  { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true } },
  { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
],
height: 265,
actionComplete: (args: DialogEditEventArgs) => {
  if ((args.requestType === 'beginEdit' || args.requestType ===
'add')) {
    let tabObj: Tab = new Tab({
      showCloseButton: false,
      selecting: (e) => {if(e.isSwiped) {e.cancel = true;}
if(e.selectingIndex === 1) {e.cancel = !validate(1)}},
      items: [
        { header: { 'text': 'Details' }, content: '#tab1' },
        { header: { 'text': 'Verify' }, content: '#tab2' },
      ]
    });
    tabObj.appendTo('#edittab');
    new DropDownList({value: args.rowData.ShipCountry, popupHeight:
'200px', floatLabelType: 'Always',
      dataSource: countryData, fields: {text: 'ShipCountry',
value: 'ShipCountry'}, placeholder: 'Ship Country'},
args.form.elements.namedItem('ShipCountry') as HTMLInputElement);
    new CheckBox({ label: 'Verified', checked: args.rowData.Verified
}, args.form.elements.namedItem('Verified'));
    // Set initail Focus
    if (args.requestType === 'beginEdit') {
      (args.form.elements.namedItem('CustomerID') as
HTMLInputElement).focus();
    }
    document.getElementById('next').onclick = () => {
      moveNext();
    }
    function validate(tab) {
      let valid: boolean = true;
      [].slice.call(document.getElementById('tab' +
tab).querySelectorAll('[name]')).forEach(element => {
        element.form.ej2_instances[0].validate(element.name);
        if (element.getAttribute('aria-invalid') === 'true') {
          valid = false;
        }
      });
      if (!valid) {
        return false;
      }
      return true;
    }
    function moveNext() {
      if (validate(1)) {
        tabObj.select(1);
      }
    }
  }
}

```

```

        document.getElementById('submit').onclick = () => {
            if (validate(2)) {
                grid.endEdit();
            }
        }
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/css/bootstrap.min.c
ss">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
    <script id='dialogtemplate' type="text/x-template">
        <div>
            <div id="edittab"></div>

```

```

        <div id="tab1">
            <div class="form-row" >
                <div class="form-group col-md-6">
                    <div class="e-float-input e-control-wrapper">
                        <input id="OrderID" name="OrderID" type="text"
value=${if(isAdd)} '' ${else} ${OrderID} ${/if} ${if(isAdd)} '' ${else}
disabled ${/if} />
                        <span class="e-float-line"></span>
                        <label class="e-float-text e-label-top"
for="OrderID">Order ID</label>
                    </div>
                </div>
            </div>
            <div class="form-row" >
                <div class="form-group col-md-6">
                    <div class="e-float-input e-control-wrapper">
                        <input id="CustomerID" name="CustomerID"
type="text" value=${if(isAdd)} '' ${else} ${CustomerID} ${/if} />
                        <span class="e-float-line"></span>
                        <label class="e-float-text e-label-top"
for="CustomerID">Customer ID</label>
                    </div>
                </div>
            </div>
            <button id='next' class='e-info e-btn' style="float: right"
type='button'> next</button>
        </div>

        <div id="tab2" style='display: none'>
            <div class="form-row" >
                <div class="form-group col-md-6">
                    <input type="text" name="ShipCountry"
id="ShipCountry" value=${if(isAdd)} '' ${else} ${ShipCountry} ${/if} />
                </div>
            </div>
            <div class="form-row">
                <div class="form-group col-md-6">
                    <input type="checkbox" name="Verified"
id="Verified" ${if(isAdd)} '' ${else} checked ${/if} />
                </div>
            </div>
            <button id='submit' class='e-info e-btn' style="float:
right" type='button'> submit</button>
        </div>

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Batch editing in EJ2 JavaScript Grid control

In Batch edit mode, when you double-click on the grid cell, then the target cell changed to edit state. You can bulk save (added, changed and deleted data in the single request) to data source by click on the toolbar's **Update** button or by externally invoking the `[batchSave]../api/grid/edit/#batchsave` method. To enable Batch edit, set the `editSettings.mode` as **Batch**.

INDEX.TS

```
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Batch' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120,
allowEditing: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
  ],
  height: 265
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If a column's [allowEditing](#) property is set to false, then the focus can be skipped in that non-editable column by clicking the tab or shift-tab key while in batch edit mode.

Automatically update the column based on another column edited value in batch mode

You can update the column value based on another column edited value in Batch mode by using the Cell Edit Template feature.

In the below demo, we have update the **TotalCost** column value based on the **UnitPrice** and **UnitInStock** column value while editing.

INDEX.TS

```
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { productData } from './productData.ts';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
Grid.Inject(Edit, Toolbar);
var priceElem: HTMLElement;;
var priceObj: NumericTextBox;
var stockElem: HTMLElement;;
var stockObj: NumericTextBox;
let grid: Grid = new Grid({
  dataSource: productData,
  editSettings: {
    allowEditing: true,
    allowAdding: true,
    allowDeleting: true,
    mode: 'Batch',
    newRowPosition: 'Top'
  },
  allowPaging: true,
  pageSettings: { pageCount: 5 },
  toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
  columns: [
    {
      field: 'ProductID',
      isPrimaryKey: true,
      headerText: 'Product ID',
      textAlign: 'Right',
      validationRules: { required: true, number: true },
      width: 140
    },
    {
      field: 'ProductName',
      headerText: 'Product Name',
      validationRules: { required: true },
      width: 140
    },
    {
      field: 'UnitPrice',
      headerText: 'UnitPrice',
      textAlign: 'Right',
      edit: {
        create: function() {
          priceElem = document.createElement('input');
          return priceElem;
        },
        read: function() {
          return priceObj.value;
        },
        destroy: function() {
          priceObj.destroy();
        },
        write: function(args) {
          var rowData = args.rowData;
```



```

        var rowIndex = grid.getRowInfo(args.row).rowIndex;
        priceObj = new NumericTextBox({
            value: args.rowData[args.column.field],
            change: function(args) {
                var totalCostValue = args.value * rowData['UnitsInStock'];
                grid.updateCell(rowIndex, 'TotalCost', totalCostValue);
            }
        });
        priceObj.appendTo(priceElem);
    },
    width: 140,
    format: 'C2',
    validationRules: { required: true }
},
{
    field: 'UnitsInStock',
    headerText: 'Units In Stock',
    textAlign: 'Right',
    edit: {
        create: function() {
            stockElem = document.createElement('input');
            return stockElem;
        },
        read: function() {
            return stockObj.value;
        },
        destroy: function() {
            stockObj.destroy();
        },
        write: function(args) {
            var rowData = args.rowData;
            var rowIndex = grid.getRowInfo(args.row).rowIndex;
            stockObj = new NumericTextBox({
                value: args.rowData[args.column.field],
                change: function(args) {
                    var totalCostValue = args.value * rowData['UnitPrice'];
                    grid.updateCell(rowIndex, 'TotalCost', totalCostValue);
                }
            });
            stockObj.appendTo(stockElem);
        }
    },
    width: 140,
    validationRules: { required: true }
},
{
    field: 'TotalCost',
    headerText: 'Total Unit Cost',
    textAlign: 'Right',
    width: 140,
    format: 'C2',
}
]
});
grid.appendTo('#Grid');
grid.cellEdit= function(args){

```

```

if(args.columnName == "TotalCost") {
    args.cancel= true;
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>

```

```

</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cancel edit based on condition in batch mode

You can prevent the CRUD operations of the Batch edit Grid by using condition in the [cellEdit](#), [beforeBatchAdd](#) and [beforeBatchDelete](#) events for Edit, Add and Delete actions respectively.

In the below demo, we prevent the CRUD operation based on the **Role** column value. If the Role Column is **Employee**, we are unable to edit/delete that row.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let isAddable: boolean = true;
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Batch' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'Role', headerText: 'Role', width: 120, },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    cellEdit: cellEdit,
    beforeBatchAdd: beforeBatchAdd,
    beforeBatchDelete: beforeBatchDelete,
    height: 240
});
grid.appendTo('#Grid');
function cellEdit(args) {
    if (args.rowData['Role'] == 'Employee') {

```

```

        args.cancel = true;
    }
}
function beforeBatchAdd(args) {
    if (!isAddable) {
        args.cancel = true;
    }
}
function beforeBatchDelete(args) {
    if (args.rowData['Role'] == 'Employee') {
        args.cancel = true;
    }
}
}
var button = document.createElement('button');
button.innerText = 'Grid is Addable';
document.body.insertBefore(button, document.body.children[0]);
button.addEventListener('click', btnClick.bind(this));
function btnClick(args) {
    args.target.innerText == 'Grid is Addable' ? (args.target.innerText =
'Grid is Not Addable') : (args.target.innerText = 'Grid is Addable');
    isAddable = !isAddable;
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Confirmation dialog

By default, grid will show the confirm dialog when saving or canceling or performing any actions like filtering, sorting, etc.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { showConfirmDialog: true, showDeleteConfirmDialog: true,
allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Batch' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },

```

```

        { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 265
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {

```

```

        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* [editSettings.showConfirmDialog](#) requires the `editSettings.mode` to be `Batch`

* If [editSettings.showConfirmDialog](#) set to `false`, then confirmation dialog does not display in batch editing.

How to make editing in single click and arrow keys

When using batch mode, the TAB key allows you to edit or move to the next cell or row from the current record by default. Using the grid's load event, the same functionality can also be achieved by pressing the arrow keys. Additionally, the `editCell` method of the grid allows for cells to be made editable with a single click.

In the following sample, the `load` event of the Grid will be used to bind the keydown event handler. When any arrow key is pressed, the `editCell` method of the Grid will be used to identify the next or previous cell (td) and set it to be editable. Additionally, it is possible to enable editing of a cell with a single click by utilizing the `editCell` method within the `created` event of the Grid.

INDEX.TS

```

import { Grid, Page, Toolbar, Edit } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { isNullOrUndefined } from '@syncfusion/ej2-base';
Grid.Inject(Page, Toolbar, Edit);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    enableHover: false,
    created: created,
    load: load,
    editSettings: {

```

```

        allowEditing: true,
        allowAdding: true,
        allowDeleting: true,
        mode: 'Batch',
    },
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true,
        visible: false, validationRules: { required: true, number: true }
    },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true }
    },
        { field: 'Freight', headerText: 'Freight', format: 'C2', width: 100,
textAlign: 'Right' },
        { field: 'OrderDate', headerText: 'Order Date', editType:
'datepickeredit', format: 'yMd', width: 100},
        { field: 'ShipCountry', headerText: 'Ship Country', width: 100 },
    ],
});
grid.appendTo('#Grid');
function created(args) {
    grid.getContentTable().addEventListener('click', function (args) {
        if ((args.target as any).classList.contains('e-rowcell')) {
            grid.editModule.editCell(parseInt((args.target as
any).getAttribute('index'))),
            grid.getColumnByIndex(parseInt(args.target.getAttribute('data-
colindex'))).field);
        }
    });
}
function load() {
    grid.element.addEventListener('keydown', function (e) {
        var closesttd = (e.target as any).closest('td');
        if (e.keyCode === 39 && !isNullOrUndefined(closesttd.nextSibling)) {
            editACell(closesttd.nextSibling);
        }
        if (e.keyCode === 37 &&
!isNullOrUndefined(closesttd.previousSibling) &&
!grid.getColumnByIndex(
            parseInt(closesttd.previousSibling.getAttribute('data-
colindex'))).isPrimaryKey)
        {
            editACell(closesttd.previousSibling);
        }
        if (e.keyCode === 40 &&
!isNullOrUndefined(closesttd.closest('tr').nextSibling)) {
            editACell(
                closesttd.closest('tr').nextSibling.querySelectorAll('td')[
                    parseInt(closesttd.getAttribute('data-colindex'))]);
        }
        if (e.keyCode === 38 &&
!isNullOrUndefined(closesttd.closest('tr').previousSibling)) {
            editACell(
                closesttd.closest('tr').previousSibling.querySelectorAll('td')[

```



```

        parseInt(closesttd.getAttribute('data-colindex'))]);
    }
});
}
function editACell(args) {
    grid.editModule.editCell(
        parseInt(args.getAttribute('index')),
        grid.getColumnByIndex(parseInt(args.getAttribute('data-
colindex'))).field);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
    </style>

```

```

        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Command column editing in EJ2 JavaScript Grid control

The command column provides an option to add CRUD action buttons in a column. This can be defined by the [column.commands](#) property.

The available built-in command buttons are:

Command Button	Actions
----- -----	
Edit	Edit the current row.
Delete	Delete the current row.
Save	Update the edited row.
Cancel	Cancel the edited state.

INDEX.TS

```

import { Grid, CommandColumn, Edit } from '@syncfusion/ej2-grids';
import { employeeData } from './datasource.ts';
Grid.Inject(CommandColumn, Edit);
let grid: Grid = new Grid({
    dataSource: employeeData,
    editSettings: { allowEditing: true, allowDeleting: true },
    columns: [
        { field: 'EmployeeID', isPrimaryKey: true, headerText: 'Employee ID', textAlign: 'Right', width: 125 },
        { field: 'FirstName', headerText: 'Name', width: 120 },
        { field: 'Title', headerText: 'Title', width: 170 },

```

```

        { headerText: 'Commands', width: 120, commands: [{ type: 'Edit',
buttonOption: { cssClass: 'e-flat', iconCss: 'e-edit e-icons' } },
        { type: 'Delete', buttonOption: { cssClass: 'e-flat', iconCss:
'e-delete e-icons' } },
        { type: 'Save', buttonOption: { cssClass: 'e-flat', iconCss: 'e-
update e-icons' } },
        { type: 'Cancel', buttonOption: { cssClass: 'e-flat', iconCss:
'e-cancel-icon e-icons' } } ] }
    ],
    height: 310
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom command column

The custom command buttons can be added in a column by using the [column.commands](#) property and the action for the custom buttons can be defined in the [commandClick](#) event.

INDEX.TS

```

import { Grid, CommandColumn, Edit, IRow, Column, CommandClickEventArgs }
from '@syncfusion/ej2-grids';
import { closest } from '@syncfusion/ej2-base';
import { employeeData } from './datasource.ts';
Grid.Inject(CommandColumn, Edit);
let grid: Grid = new Grid({
  dataSource: employeeData,
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 125 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'Title', headerText: 'Title', width: 170 },
    { headerText: 'Commands', width: 120, commands: [{ buttonOption: {
content: 'Details', cssClass: 'e-flat' } } ] },
  ],
  commandClick: (args: CommandClickEventArgs) => {
    alert(JSON.stringify(args.rowData));
  },
  height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Validation in EJ2 JavaScript Grid control

Column validation

Column validation allows you to validate the edited or added row data and it display errors for invalid fields before saving data.

Grid uses **Form Validator** component for column validation. You can set validation rules by defining the [columns.validationRules](#).

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);

```

```

let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true } },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true, minLength: 3 } },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 265
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {

```

```

        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom validation

You can define your own custom validation rules for the specific columns by using **Form Validator** custom rules.

In the below demo, custom validation applied for **CustomerID** column.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let customFn: (args: { [key: string]: string }) => boolean = (args: { [key:
string]: string }) => {
    return args['value'].length >= 5;
};
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
    columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true } },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true, minLength: [customFn, 'Need at least 5
letters'] } },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 265
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
  </style>

```



```

        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom validation based on dropdown change

You can apply validation rules and messages to a column based on another column value in edit mode. You can achieve this requirement by using the custom validation feature of Grid.

In the following sample, dropdownlist edit type is used for the **Role** and **Salary** columns. Here, you can apply the custom validation in the **Salary** column based on the value selected in the **Role** column.

INDEX.TS

```

import { Grid, Page, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { employeeDetails } from './datasource.ts';
import { Query } from '@syncfusion/ej2-data';
Grid.Inject(Page, Edit, Toolbar);
let jobRole: { [key: string]: Object }[] = [
    { role: 'TeamLead', destId: '0' },
    { role: 'Manager', destId: '1' },
    { role: 'Engineer', destId: '2' },
    { role: 'Sales', destId: '3' },
    { role: 'Support', destId: '4' },
];
let salaryDetails: { [key: string]: Object }[] = [
    { salary: '11000', destId: '1' },
    { salary: '13500', destId: '2' },
    { salary: '16500', destId: '2' },
    { salary: '18500', destId: '1' },
    { salary: '21500', destId: '2' },
];

```

```

        { salary: '23000', destId: '2' },
    ];
    let grid: Grid = new Grid({
        dataSource: employeeDetails,
        allowPaging: true,
        toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
        editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
        load: () => {
            var column = grid.getColumnByField('Salary');
            column.validationRules = {
                required: [customFn, window['Please enter valid salary']],
            };
        },
        columns: [
            {
                field: 'EmployeeID',
                headerText: 'Employee ID',
                isPrimaryKey: true,
                textAlign: 'Right',
                width: 120
            },
            {
                field: 'Role',
                headerText: 'Role',
                width: 120,
                editType: 'dropdownedit',
                edit: {
                    params: {
                        query: new Query(),
                        dataSource: jobRole,
                        fields: { value: 'role', text: 'role' },
                        allowFiltering: true,
                        change: valChange
                    }
                }
            },
            {
                field: 'Salary',
                headerText: 'Salary',
                width: 160,
                editType: 'dropdownedit',
                edit: {
                    params: {
                        query: new Query(),
                        dataSource: salaryDetails,
                        fields: { value: 'salary', text: 'salary' },
                        allowFiltering: true
                    }
                }
            },
            {
                field: 'Address',
                headerText: 'Address',
                width: 160
            }
        ]
    });

```

```

});
grid.appendTo('#Grid');
(window as any).role = '';
function customFn(args) {
    switch (window['role']) {
        case 'Engineer':
            if (args.value > 10000 && args.value < 15000) {
                return true;
            } else {
                this.rules['Salary']['required'][1] = 'Please enter valid
Engineer Salary';
            }
            break;
        case 'TeamLead':
            if (args.value > 15000 && args.value < 20000) {
                return true;
            } else {
                this.rules['Salary']['required'][1] = 'Please enter valid
TeamLead Salary';
            }
            break;
        case 'Manager':
            if (args.value > 20000 && args.value < 25000) {
                return true;
            } else {
                this.rules['Salary']['required'][1] = 'Please enter valid
Manager Salary';
            }
            break;
        case 'Sales':
            if (args.value > 5000 && args.value < 25000) {
                return true;
            } else {
                this.rules['Salary']['required'][1] = 'Please enter valid
Manager Salary';
            }
            break;
        case 'Support':
            if (args.value > 10000 && args.value < 19000) {
                return true;
            } else {
                this.rules['Salary']['required'][1] = 'Please enter valid
Manager Salary';
            }
            break;
    }
    return false;
}
function valChange(args) {
    window['role'] = args.value;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
.e-row[aria-selected="true"] .e-customizedExpandcell {
    background-color: #e0e0e0;
}
.e-grid.e-gridhover tr[role='row']:hover {
    background-color: #eee;
}
.e-expand::before {
    content: '\e5b8';
}
.empImage {
    margin: 6px 16px;
    float: left;
    width: 50px;
    height: 50px;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

<div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Persisting data in server in EJ2 JavaScript Grid control

Edited data can be persisted in the database using the RESTful web services.

All the CRUD operations in the grid are done through [DataManager](#). The `DataManager` has an option to bind all the CRUD related data in server-side.

For your information, the `ODataAdaptor` persists data in the server as per OData protocol.

In the below section, we have explained how to get the edited data details on the server-side using the [UrlAdaptor](#).

Using URL adaptor

You can use the [UrlAdaptor](#) of `DataManager` when binding data source from remote data. In the initial load of grid, data are fetched from remote data and bound to the grid using `url` property of `DataManager`. You can map The CRUD operation in grid can be mapped to server-side Controller actions using the properties `insertUrl`, `removeUrl`, `updateUrl`, `crudUrl` and `batchUrl`.

The following code example describes the above behavior.

```

`ts
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';

Grid.Inject(Edit, Toolbar);

let data: DataManager = new DataManager({
    url: "Home/DataSource",
    updateUrl: "Home/Update",
    insertUrl: "Home/Insert",
    removeUrl: "Home/Delete",
    adaptor: new UrlAdaptor
});

let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Normal' },

```

```

columns: [
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100, isPrimaryKey: true,
validationRules: { required: true } },
{ field: 'CustomerID', headerText: 'Customer ID', width: 120, validationRules: { required: true, minLength:
3 }, defaultValue: 'HANAR' },
{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', editType: 'numericedit', width: 120, format: 'C2'
},
{ field: 'ShipCountry', headerText: 'Ship Country', editType: 'dropdownedit', width: 150 }
],
height: 265
});
grid.appendTo('#Grid');
`

```

Also, when using the `UrlAdaptor`, you need to return the data as JSON from the controller action and the JSON object must contain a property as `result` with `dataSource` as its value and one more property `count` with the `dataSource` total records count as its value.

The following code example describes the above behavior.

```

`ts
public ActionResult DataSource(DataManager dm)
{
var DataSource = OrderRepository.GetAllRecords();
DataResult result = new DataResult();
result.result = DataSource.Skip(dm.Skip).Take(dm.Take).ToList();
result.count = result.result.Count;
return Json(result, JsonRequestBehavior.AllowGet);
}
public class DataResult
{
public List<EditableOrder> result { get; set; }
public int count { get; set; }
}
`

```

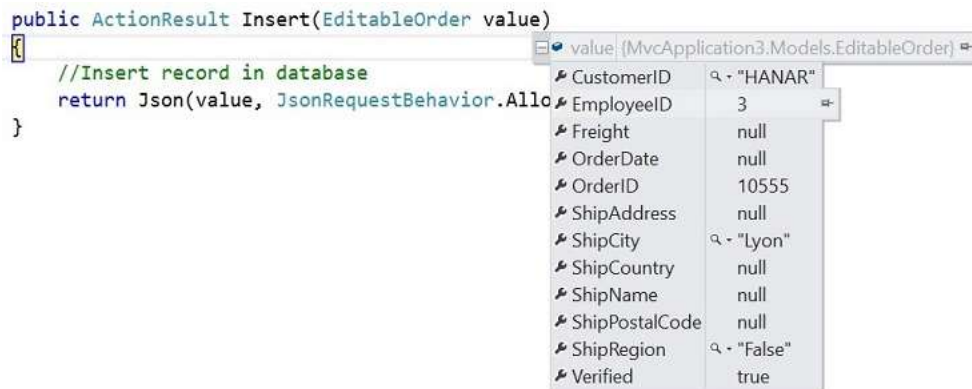
Insert record

Using the `insertUrl` property, you can specify the controller action mapping URL to perform insert operation on the server-side.

The following code example describes the above behavior.

```
`ts
public ActionResult Insert(EditableOrder value)
{
    //Insert record in database
}
`
```

The newly added record details are bound to the `value` parameter. Please refer to the following screenshot.



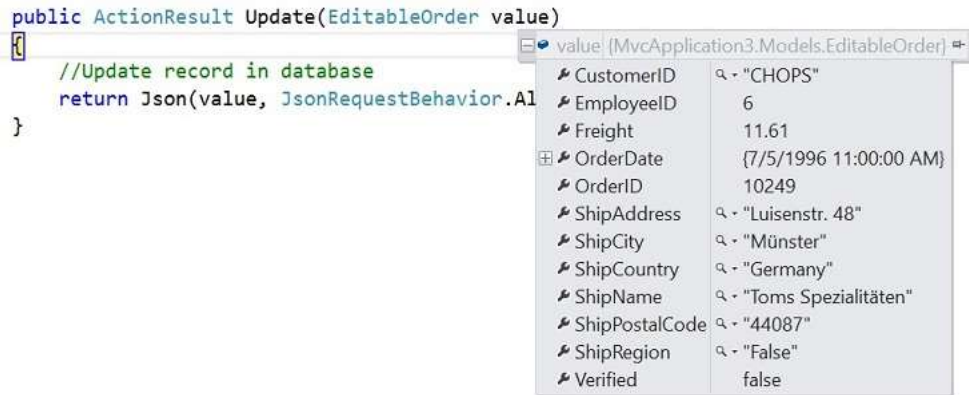
Update record

Using the `updateUrl` property, the controller action mapping URL can be specified to perform save/update operation on the server-side.

The following code example describes the previous behavior.

```
`ts
public ActionResult Update(EditableOrder value)
{
    //Update record in database
}
`
```

The updated record details are bound to the `value` parameter. Please refer to the following screenshot.



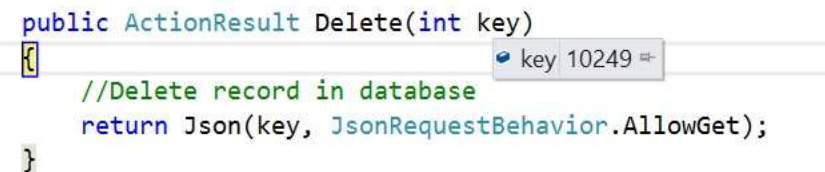
Delete record

Using the `removeUrl` property, the controller action mapping URL can be specified to perform delete operation on the server-side.

The following code example describes the previous behavior.

```
`ts
public ActionResult Delete(int key)
{
    //Delete record in database
}
```

The deleted record primary key value is bound to the `key` parameter. Please refer to the following screenshot.



CRUD URL

Using the `crudUrl` property, the controller action mapping URL can be specified to perform all the CRUD operation at server-side using a single method instead of specifying separate controller action method for CRUD (insert, update and delete) operations.

The action parameter of `crudUrl` is used to get the corresponding CRUD action.

The following code example describes the above behavior.

```
`ts
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Edit, Toolbar);
```



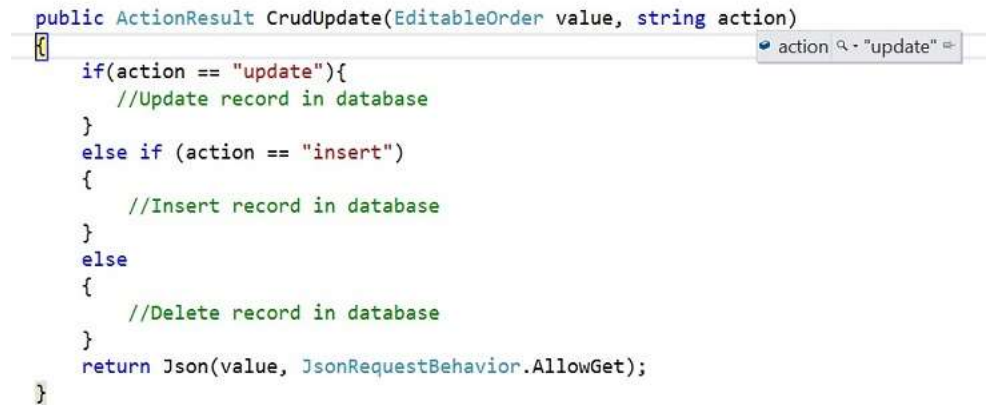
```

let data: DataManager = new DataManager({
url: "Home/DataSource",
crudUrl: "Home/CrudUpdate",
adaptor: new UrlAdaptor
});
let grid: Grid = new Grid({
dataSource: data,
toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Normal' },
columns: [
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100, isPrimaryKey: true,
validationRules: { required: true } },
{ field: 'CustomerID', headerText: 'Customer ID', width: 120, validationRules: { required: true, minLength:
3 }, defaultValue: 'HANAR' },
{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', editType: 'numericedit', width: 120, format: 'C2'
},
{ field: 'ShipCountry', headerText: 'Ship Country', editType: 'dropdownedit', width: 150 }
],
height: 265
});
grid.appendTo('#Grid');
`ts
public ActionResult CrudUpdate(EditableOrder value, string action)
{
if(action == "update"){
//Update record in database
}
else if (action == "insert")
{
//Insert record in database
}
else
{

```

```
//Delete record in database
}
}
,
```

Please refer to the following screenshot to know about the action parameter.



```
public ActionResult CrudUpdate(EditableOrder value, string action)
{
    if(action == "update"){
        //Update record in database
    }
    else if (action == "insert")
    {
        //Insert record in database
    }
    else
    {
        //Delete record in database
    }
    return Json(value, JsonRequestBehavior.AllowGet);
}
```

If you specify `insertUrl` along with `crudUrl`, then while adding `insertUrl` only will be invoked.

Batch URL

The `batchUrl` property supports only for batch editing mode. You can specify the controller action mapping URL to perform batch operation on the server-side.

The following code example describes the above behavior.

```
`ts
import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Edit, Toolbar);
let data: DataManager = new DataManager({
    url: "Home/DataSource",
    batchUrl: "Home/BatchUpdate",
    adaptor: new UrlAdaptor
});
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Batch' },
    columns: [
```

```

{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100, isPrimaryKey: true,
validationRules: { required: true } },

{ field: 'CustomerID', headerText: 'Customer ID', width: 120, validationRules: { required: true, minLength:
3 }, defaultValue: 'HANAR' },

{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', editType: 'numericedit', width: 120, format: 'C2'
},

{ field: 'ShipCountry', headerText: 'Ship Country', editType: 'dropdownedit', width: 150 }
],
height: 265
});
grid.appendTo('#Grid');
`
`ts

```

```

public ActionResult BatchUpdate(string action, List<EditableOrder> added, List<EditableOrder> changed,
List<EditableOrder> deleted, int? key)

```

```

{
//Save the batch changes in database
}

```

```

public ActionResult BatchUpdate(string action, List<EditableOrder> added, List<EditableOrder> changed, List<EditableOrder> deleted, int? key)
{
//Save the batch changes in database
return Json(changed, JsonRequestBehavior.AllowGet);
}

```

changed: Count = 2

CustomerID	"HANAR-Updated"
EmployeeID	4
Freight	65.83
OrderDate	(7/8/1996 5:30:00 AM)
OrderID	10250
ShipAddress	"Rua do Paço, 67"
ShipCity	"Rio de Janeiro"
ShipCountry	"Brazil"
ShipName	"Hanari Carnes"
ShipPostalCode	"05454-876"
ShipRegion	"RJ"
Verified	true

Sorting in EJ2 JavaScript Grid control

Sorting enables you to sort data in the **Ascending** or **Descending** order. To sort a column, click the column header.

To sort multiple columns, press and hold the CTRL key and click the column header. You can clear sorting of any one of the multi-sorted columns by pressing and holding the SHIFT key and clicking the specific column header.

To enable sorting in the Grid, set the [allowSorting](#) to true. Sorting options can be configured through the [sortSettings](#).

To sort, inject the [Sort](#) module in the grid.

INDEX.TS

```

import { Grid, Sort } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';

```

```

Grid.Inject(Sort);
let grid: Grid = new Grid({
    dataSource: data,
    allowSorting: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
    </style>

```

```

    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* Grid columns are sorted in the **Ascending** order. If you click the already sorted column, the sort direction toggles.

* You can apply and clear sorting by invoking [sortColumn](#) and [clearSorting](#) methods.

* To disable sorting for a particular column, set the [columns.allowSorting](#) to false.

Initial sort

To sort at initial rendering, set the [field](#) and [direction](#) in the **sortSettings.columns**.

INDEX.TS

```

import { Grid, Sort } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Sort);
let grid: Grid = new Grid({
    dataSource: data,
    allowSorting: true,
    sortSettings: { columns: [{ field: 'OrderID', direction: 'Ascending' },
    { field: 'ShipCity', direction: 'Descending' }] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },

```

```

        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>

```

```

    }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multi-column sorting

You can sort more than one column in a Grid. To sort multiple columns, press and hold the **CTRL** key and click the column header. The sorting order will be displayed in the header while performing multi-column sorting.

To clear sorting for a particular column, press the "Shift + mouse left click".

The [allowSorting](#) must be true while enabling multi-column sort.

Set [allowMultiSorting](#) property as **false** to disable multi-column sorting.

INDEX.TS

```

import { Grid, Sort } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Sort);
let grid: Grid = new Grid({
    dataSource: data,
    allowSorting: true,
    allowMultiSorting: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

<div id="container">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sort order

By default, the sorting order will be as **ascending -> descending -> none**.

When first click a column header it sorts the column in ascending. Again click the same column header, it will sort the column in descending. A repetitive third click on the same column header will clear the sorting.

Sort foreign key column based on Text

For local data in Grid, sorting will be performed based on the [foreignKeyValue](#).

For remote data in Grid, sorting will be performed based on the [foreignKeyField](#), we need to handle the sorting operation at the server side.

```

`ts
import { Grid, ForeignKey, Sort } from '@syncfusion/ej2-grids';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
Grid.Inject(ForeignKey, Sort);
let dataSource: DataManager = new DataManager({
  json: '/OData/Items',
  adaptor: new ODataV4Adaptor
});
let employeeData: DataManager = new DataManager({
  url: '/OData/Brands',
  adaptor: new ODataV4Adaptor
});
let grid: Grid = new Grid(
{
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100 },
    {

```

```

field: 'EmployeeID', headerText: 'Employee Name', width: 150, foreignKeyValue: 'FirstName',
dataSource: employeeData
},
{ field: 'Freight', headerText: 'Freight', width: 100, textAlign: 'Right'},
{ field: 'ShipName', headerText: 'Ship Name', width: 180 }
],
height: 315
});
grid.appendTo('#Grid');
`

```

The following code example describes the handling of sorting operation at the server side.

```

public class ItemsController : ODataController
{
    [EnableQuery]
    public IQueryable<Item> Get()
    {
        List<Item> GridData =
        JsonConvert.DeserializeObject<Item[]>(Properties.Resources.ItemsJson).AsQueryable().ToList();
        List<Brand> empData =
        JsonConvert.DeserializeObject<Brand[]>(Properties.Resources.BrandsJson).AsQueryable().ToList();
        var queryString = HttpContext.Current.Request.QueryString;
        var allUrlKeyValues = ControllerContext.Request.GetQueryNameValuePairs();
        string key = allUrlKeyValues.LastOrDefault(x => x.Key == "$orderby").Value;
        if (key != null)
        {
            if (key == "EmployeeID") {
                GridData = SortFor(key); //Only for foreignKey Column ascending
            }
            else if(key == "EmployeeID desc") {
                GridData = SortFor(key); //Only for foreignKey Column descending
            }
        }
        var count = GridData.Count();
    }
}

```

```

var data = GridData.AsQueryable();
return data;
}
public List<Item> SortFor(String Sorted)
{
    List<Item> GridData =
    JsonConvert.DeserializeObject<Item[]>(Properties.Resources.ItemsJson).AsQueryable().ToList();
    List<Brand> empData =
    JsonConvert.DeserializeObject<Brand[]>(Properties.Resources.BrandsJson).AsQueryable().ToList();
    if (Sorted == "EmployeeID") //check whether ascending or descending
    empData = empData.OrderBy(e => e.FirstName).ToList();
    else if(Sorted == "EmployeeID desc")
    empData = empData.OrderByDescending(e => e.FirstName).ToList();
    List<Item> or = new List<Item>();
    for (int i = 0; i < empData.Count(); i++) {
        //Select the Field matching records
        IEnumerable<Item> list = GridData.Where(pred => pred.EmployeeID ==
        empData[i].EmployeeID).ToList();
        or.AddRange(list);
    }
    return or;
}
}
,

```

Sorting events

During the sort action, the grid component triggers two events. The [actionBegin](#) event triggers before the sort action starts, and the [actionComplete](#) event triggers after the sort action is completed. Using these events you can perform the needed actions.

INDEX.TS

```

import { Grid, Sort, SortEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Sort);
let grid: Grid = new Grid({
    dataSource: data,
    allowSorting: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },

```

```

        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315,
    actionBegin: actionHandler,
    actionComplete: actionHandler
});
grid.appendTo('#Grid');
function actionHandler(args: SortEventArgs) {
    alert(args.requestType + ' ' + args.type); //custom Action
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The [args.requestType](#) is the current action name. For example, in sorting the [args.requestType](#) value is 'sorting'.

Custom sort comparer

You can customize the default sort action for a column by defining the [column.sortComparer](#) property. The sort comparer function has the same functionality like [Array.sort](#) sort comparer.

In the following example, custom sort comparer function was defined in the **Customer ID** column.

INDEX.TS

```

import { Grid, Sort } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Sort);
// The custom function
let sortComparer: (reference: string, comparer: string) => number =
(reference: string,
comparer: string) => {
  if (reference < comparer) {
    return -1;
  }
  if (reference > comparer) {
    return 1;
  }
  return 0;
};
let grid: Grid = new Grid({
  dataSource: data,
  allowSorting: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120 },
    { field: 'CustomerID', sortComparer: sortComparer, width:
140, headerText: 'Customer ID' },
    { field: 'Freight', textAlign: 'Right', width: 120, format:
'C2' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign:
'Right', width: 140, format: 'yMd' }
  ],
  height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">



  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="dropDown"></div>
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

The sort comparer function will work only for the local data.

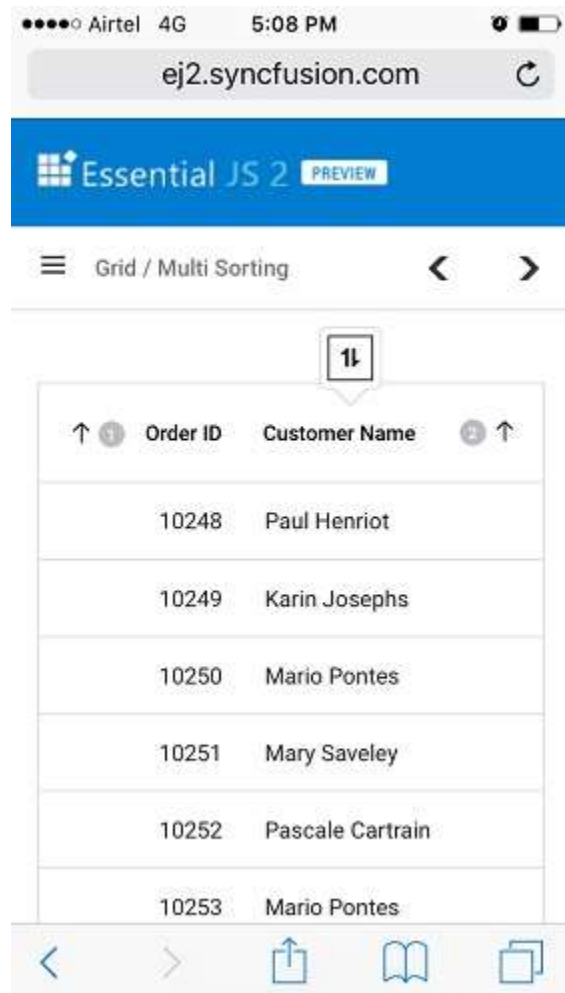
Touch interaction

When you tap the grid header on touchscreen devices, the selected column header is sorted. A popup

 is displayed for multi-column sorting. To sort multiple columns, tap the popup , and then tap the desired grid headers.

The [allowMultiSorting](#) and [allowSorting](#) should be `true` then only the popup will be shown.

The following screenshot shows grid touch sorting.



Grouping

Grouping in EJ2 JavaScript Grid control

The Grid has options to group records by dragging and dropping the column header to the group drop area. When grouping is applied, grid records are organized into a hierarchical structure to facilitate easier expansion and collapse of records.

To enable grouping in the grid, set the [allowGrouping](#) as true. Grouping options can be configured through the [groupSettings](#). To group, inject the [Group](#) module in the grid.

INDEX.TS

```
import { Grid, Group } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group);
let grid: Grid = new Grid({
  dataSource: data,
  allowGrouping: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 267
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
```



```

        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* You can group and ungroup columns by using the [groupColumn](#) and [ungroupColumn](#) methods.

* To disable grouping for a particular column, set the [columns.allowGrouping](#) to false.

Initial group

To apply group at initial rendering, set the column field name in the `groupSettings.columns`.

INDEX.TS

```

import { Grid, Group } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group);
let grid: Grid = new Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { columns: ['CustomerID', 'ShipCity'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
});

```

```

    ],
    height: 267
  });
  grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>

```

```

</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hide drop area

To avoid ungrouping or further grouping of a column after initial column grouping, define the [groupSettings.showDropArea](#) as false.

INDEX.TS

```

import { Grid, Group } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group);
let grid: Grid = new Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { showDropArea: false, columns: ['CustomerID',
'ShipCity'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

        ele.style.visibility = "visible";
    }
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Group with paging

On grouping columns with paging feature, the aggregated information and total items are displayed based on the current page. The grid does not consider aggregated information and total items from other pages. To get additional details (aggregated information and total items) from other pages, set the [groupSettings.disablePageWiseAggregates](#) to false.

If remote data is bound to grid dataSource, two requests will be sent when performing grouping action; one for getting the grouped data and another for getting aggregate details and total items count.

Group by format

By default, a column will be grouped by the data or value present for the particular row. To group the numeric or datetime column based on the mentioned format, you have to enable the [enableGroupByFormat](#) property of the corresponding

grid columns.

INDEX.TS

```

import { Grid, Group } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group);
let grid: Grid = new Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { showDropArea: false, columns: ['OrderDate', 'Freight']
},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'OrderDate', headerText: 'Order Date', format: 'yMMM',
enableGroupByFormat: true, width: 150 },
        { field: 'Freight', headerText: 'Freight', format: 'C2',
enableGroupByFormat: true, width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouping events

During the group action, the grid component triggers two events. The [actionBegin](#) event triggers before the group action starts and the [actionComplete](#) event triggers after the group action is completed. Using these events you can perform any action.

INDEX.TS

```

import { Grid, Group, ActionEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group);
let grid: Grid = new Grid({
  dataSource: data,
  allowGrouping: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 260,
  actionBegin: actionHandler,
  actionComplete: actionHandler
});
grid.appendTo('#Grid');
function actionHandler(args: ActionEventArgs) {
  alert(args.requestType + ' ' + args.type); //Custom Action
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The [args.requestType](#) is based on the current action name. For example, when grouping, the [args.requestType](#) value will be 'grouping'.

Collapse by external button

You can collapse the selected group from an external button by invoking the [expandCollapseRows](#) method.

INDEX.TS

```

import { Grid, Group } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
Grid.Inject(Group);
let grid: Grid = new Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { columns: ['ShipCity'] },
    columns: [

```



```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 240
});
grid.appendTo('#Grid');
let collapse = Button({ cssClass: 'e-flat' }, '#collapse');
document.getElementById('collapse').addEventListener('click', () => {
    if (grid.getSelectedRowIndex().length) {
        let currentTr: Element =
grid.getRows()[grid.getSelectedRowIndex()[0]];
        while (currentTr.classList && currentTr.classList.length) {
            currentTr = <Element>currentTr.previousSibling;
        }
        let collapseElement = currentTr.querySelector('.e-
recordplusexpend');
        grid.groupModule.expandCollapseRows(collapseElement); //Pass the
collapse row element.
    }
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="collapse">Collapse</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sort grouped columns in descending order during initial grouping

By default, grouped columns are sorted in ascending order. To sort grouped columns in descending order during initial grouping, you can set the [field](#) and [direction](#) in the `sortSettings.columns` property.

The `CustomerID` column will be sorted in descending order when the grid is initially grouped, as shown in the following example.

INDEX.TS

```

import { Grid, Group, Sort } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group, Sort);
let grid: Grid = new Grid({
    dataSource: data,
    allowGrouping: true,
    allowSorting: true,
    sortSettings: { columns: [{ field: 'CustomerID', direction: 'Descending'
} ] },
    groupSettings: { columns: ['CustomerID'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 267
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Exporting grouped records](#)
- [How to apply formatting for the group caption template](#)
- [How to hide expand/collapse icon for groups with single item](#)

Caption template in EJ2 JavaScript Grid control

You can customize the group caption by using the [groupSettings.captionTemplate](#) property.

INDEX.TS

```

import { Grid, Group } from '@syncfusion/ej2-grids';
import { employeeData } from './datasource.ts';
Grid.Inject(Group);
let grid: Grid = new Grid({
    dataSource: employeeData,
    allowGrouping: true,
    groupSettings: { captionTemplate: '#captiontemplate', columns:
['EmployeeID'] },
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID' },
        { field: 'CustomerID', headerText: 'Customer ID' },
        { field: 'FirstName', headerText: 'Name', width: 120 },
        { field: 'Title', headerText: 'Title', width: 170 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="captiontemplate" type="text/x-template">
            ${field} - ${key}
        </script>
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding custom text in group caption

You can customize the group caption text by using the [groupSettings.captionTemplate](#) property.

INDEX.TS

```
import { Grid, Group } from '@syncfusion/ej2-grids';
```

```
import { employeeData } from './datasource.ts';
Grid.Inject(Group);
let grid: Grid = new Grid({
    dataSource: employeeData,
    allowGrouping: true,
    groupSettings: { captionTemplate: '#captiontemplate', columns:
['EmployeeID'] },
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID' },
        { field: 'CustomerID', headerText: 'Customer ID' },
        { field: 'FirstName', headerText: 'Name', width: 120 },
        { field: 'Title', headerText: 'Title', width: 170 }
    ],
    height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="captiontemplate" type="text/x-template">
            <div>${template(data)}</div>
        </script>
        <div id="Grid"></div>
    </div>
    <script type="text/javascript">
        function template(args) {
            //you can add add customtext here.
            return args.key + " :" + args.count + " custom text" ;
        }
    </script>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Render custom component in group caption

We can render custom components in the group caption by using the [groupSettings.captionTemplate](#) property.

In the below demo, we have rendered the EJ2 [ButtonComponent](#) in the group caption.

INDEX.TS

```

import { Grid, Group, Page } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from './datasource.ts';
Grid.Inject(Group, Page);
let grid: Grid = new Grid(
    {
        dataSource: data,
        allowPaging: true,
        dataBound: () => {
            let groupCations: HTMLCollection =
document.getElementsByClassName('groupbutton');
            for(var i=0; i<groupCations.length; i++) {
                var button = new Button({
                    isPrimary: true
                });
                button.appendTo(groupCations[i] as HTMLElement);
            }
        },
        height: 315
        allowGrouping: true,
        groupSettings: { columns: ["ShipCountry"], captionTemplate:
"#captiontemplate"},

```

```

        columns: [
            { field: 'OrderID', headerText: 'Order ID', width: 120,
textAlign: 'Right' },
            { field: 'CustomerName', headerText: 'Customer Name', width:
150 },
            { field: 'OrderDate', headerText: 'Order Date', width: 130,
format: 'yMd', textAlign: 'Right' },
            { field: 'Freight', width: 120, format: 'C2', textAlign:
'Right' },
            { field: 'ShippedDate', headerText: 'Shipped Date', width:
140, format: 'yMd', textAlign: 'Right' },
            { field: 'ShipCountry', headerText: 'Ship Country', width:
150 }
        ]
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
    <script id="captiontemplate" type="text/x-template">
        ${field} - ${key} - <button class='groupbutton'>${count}
items</button>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Reordering on grouped columns in EJ2 JavaScript Grid control

Grid provides an option to interchange the position of the Grouped Columns by dragging and dropping the grouped headercells in the droparea. So, any new column entering into the droparea can also be dropped in any position.

To enable this feature, you have to set the [groupSettings.allowReordering](#) property as **true**.

INDEX.TS

```

import { Grid, Group } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
Grid.Inject(Group);
let grid: Grid = new Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { columns: ['ShipCity'], allowReordering: true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 240
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Lazy load grouping in EJ2 JavaScript Grid control

The lazy load grouping allows you to load grouped records to the Grid through the on-demand concept. So, you can use this feature to load a huge amount of grouped data to the Grid without any performance degradation.

When you enable this feature, the Grid will render only the initial level caption rows in the collapsed state at grouping. The child rows of each caption will be fetched from the server and render in the Grid when you expand the caption row. The fetching child records count will be implicitly determined by the content area occupying rows count. So, if the child records exceed the count, then the Grid will request the server again to fetch the next block of child records on scrolling.

The caption row expand/collapse state will be persisted on paging and Grid pages count will be determined based on the caption rows count instead of the child rows.

To enable this feature, you have to set the [groupSettings.enableLazyLoading](#) property as **true**.

INDEX.TS

```
import { Grid, Page, Group, LazyLoadGroup } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Group, LazyLoadGroup);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowGrouping: true,
    groupSettings: { enableLazyLoading: true, columns: ['ProductName',
'CustomerName'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'ProductName', headerText: 'Product Name', width: 160 },
        { field: 'ProductID', headerText: 'Product ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'CustomerName', headerText: 'Customer Name', width: 160 }
    ]
    height: 240
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Handling the lazy load grouping at server-side

You can use the [UrlAdaptor](#) of [DataManager](#) when binding the remote data. Along with the default server request, this feature will additionally send the below details to handle the lazy load grouping. In the server end, these details are bound with the [ISLazyLoad](#) and [OnDemandGroupInfo](#) parameters in the [DataManagerRequest](#) model. Please refer to the below table and screenshots.

Property Name | Description

[isLazyLoad](#) | To differentiate the default grouping and lazy load grouping

[onDemandGroupInfo](#) | Have the details of expanded caption row grouping [level](#), [skip](#), [take](#) and [filter](#) query of the child records

```

if (dm.IsLazyLoad)
{
    dm.IsLazyLoad = true;
    groupedData = operation.PerformGroupingCustomers(dataSource, dm); // Lazy load grouping
    if (dm.OnDemandGroupInfo != null && dm.Group.Count() == dm.OnDemandGroupInfo.Level)
    {
        count = groupedData.Cast<Customers>().Count();
    }
    else
    {
        count = groupedData.Cast<Group>().Count();
    }
    groupedData = operation.PerformSkip(groupedData, dm.OnDemandGroupInfo.Skip, dm.OnDemandGroupInfo.Skip);
    groupedData = operation.PerformTake(groupedData, dm.OnDemandGroupInfo.Take, dm.OnDemandGroupInfo.Take);
    return dm.RequiresCounts ? Json(new { result = groupedData == null ? DataSource : groupedData, count = count }) : Json(DataSource);
}

```

```

.....if (dm.IsLazyLoad)
.....{
.....    groupedData = operation.PerformGrouping<Customers>(DataSource, dm); // Lazy load grouping
.....    if (dm.OnDemandGroupInfo != null && dm.Group.Count() == dm.OnDemandGroupInfo.Level)
.....    {
.....        count = groupedData.Count();
.....        dm.OnDemandGroupInfo = (Syncfusion.EJ2.Base.OnDemandGroupInfo) new OnDemandGroupInfo
.....        {
.....            Level: 1,
.....            Skip: 0,
.....            Take: 33,
.....            Where: Count = 1
.....        };
.....    }
.....    else
.....    {
.....        count = groupedData.Count();
.....    }
.....    groupedData = operation.PerformSkip(groupedData, dm.OnDemandGroupInfo == null ? dm.Skip : dm.OnDemandGroupInfo.Skip);
.....    groupedData = operation.PerformTake(groupedData, dm.OnDemandGroupInfo == null ? dm.Take : dm.OnDemandGroupInfo.Take);
.....    return dm.RequiresCounts ? Json(new { result = groupedData == null ? DataSource : groupedData, count = count }) : Json(DataSource);
.....}

```

The following code example describes the lazy load grouping handled at the server-side with other grid actions.

```
`ts
```

```

public IActionResult UrlDatasource([FromBody] DataManagerRequest dm)
{
    IEnumerable<Customers> groupedData = null;
    IEnumerable<Customers> DataSource = customers;
    DataOperations operation = new DataOperations();
    if (dm.Search != null && dm.Search.Count > 0)
    {
        DataSource = operation.PerformSearching(DataSource, dm.Search); //Search
    }
    if (dm.Sorted != null && dm.Sorted.Count > 0) //Sorting
    {
        DataSource = operation.PerformSorting(DataSource, dm.Sorted);
    }
    if (dm.Where != null && dm.Where.Count > 0) //Filtering
    {
        DataSource = operation.PerformFiltering(DataSource, dm.Where, dm.Where[0].Operator);
    }
    int count = DataSource.Cast<Customers>().Count();
    if (dm.IsLazyLoad == false && dm.Skip != 0)
    {
        DataSource = operation.PerformSkip(DataSource, dm.Skip); // Paging
    }
    if (dm.IsLazyLoad == false && dm.Take != 0)
    {
        DataSource = operation.PerformTake(DataSource, dm.Take);
    }
}

```

```

}
if (dm.IsLazyLoad)
{
groupedData = operation.PerformGrouping<Customers>(DataSource, dm); // Lazy load grouping
if (dm.OnDemandGroupInfo != null && dm.Group.Count() == dm.OnDemandGroupInfo.Level)
{
count = groupedData.Cast<Customers>().Count();
}
else
{
count = groupedData.Cast<Group>().Count();
}

groupedData = operation.PerformSkip(groupedData, dm.OnDemandGroupInfo == null ? dm.Skip :
dm.OnDemandGroupInfo.Skip);

groupedData = operation.PerformTake(groupedData, dm.OnDemandGroupInfo == null ? dm.Take :
dm.OnDemandGroupInfo.Take);
}

return dm.RequiresCounts ? Json(new { result = groupedData == null ? DataSource : groupedData, count
= count }) : Json(DataSource);
}
,

```

Lazy load grouping with infinite scrolling

Infinite scrolling loads a huge amount of data without degrading the Grid's performance. By default, infinite scrolling is enabled only for the expanded grouped rows when lazy loading is enabled. Now, the Grid has an option to allow infinite scrolling for group caption rows. This is achieved by setting the [enableInfiniteScrolling](#) property as true when lazy loading is enabled in the grouped records.

This is demonstrated in the following sample:

INDEX.TS

```

import { Grid, Group, LazyLoadGroup, InfiniteScroll } from '@syncfusion/ej2-
grids';
import { data } from './datasource.ts';
Grid.Inject(Group, LazyLoadGroup, InfiniteScroll);
let grid: Grid = new Grid({
  dataSource: data,
  allowGrouping: true,
  enableInfiniteScrolling: true,
  groupSettings: { enableLazyLoading: true, columns: ['ProductName',
'CustomerName'] },
  columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'ProductName', headerText: 'Product Name', width: 160 },
        { field: 'ProductID', headerText: 'Product ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'CustomerName', headerText: 'Customer Name', width: 160 }
    ]
    height: 240
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The Grid also supports the lazy load grouping with [virtualization](#)(virtual scrolling).

Limitations for lazy load grouping

- Due to the element height limitation in browsers, the maximum number of records loaded by the grid is limited due to the browser capability.
- DataManager's `UrlAdaptor` and `JsonAdaptor` only have the built-in lazy load grouping support.
- Lazy load grouping is not compatible with batch editing, row template etc.
- Programmatic selection is not supported.

Filtering

Filtering in EJ2 JavaScript Grid control

Filtering allows you to view particular records based on filter criteria. To enable filtering in the Grid, set the [allowFiltering](#) to true. Filtering options can be configured through [filterSettings](#).

To use filter, inject the [Filter](#) module in the grid.

INDEX.TS

```

import { Grid, Filter } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Filter);
let grid: Grid = new Grid({
  dataSource: data,
  allowFiltering: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```



```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

* You can apply and clear filtering by using [filterByColumn](#) and [clearFiltering](#) methods.

* To disable filtering for a particular column, set [columns.allowFiltering](#) to false.

Initial filter

To apply the filter at initial rendering, set the filter [predicate](#) object in [filterSettings.columns](#).

INDEX.TS

```
import { Grid, Filter } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Filter);
let grid: Grid = new Grid({
  dataSource: data,
  allowFiltering: true,
  filterSettings: {
    columns: [{ field: 'ShipCity', matchCase: false, operator:
'startswith', predicate: 'and', value: 'reims' },
    { field: 'ShipName', matchCase: false, operator: 'startswith',
predicate: 'and', value: 'Vins et alcools Chevalier' }]
  },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 273
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter operators

The filter operator for a column can be defined in the [filterSettings.columns.operator](#) property.

The available operators and its supported data types are:

Operator | Description | Supported Types

a*b | Everything that starts with "a" and ends with "b".

a* | Everything that starts with "a".

*b | Everything that ends with "b".

a | Everything that has an "a" in it.

ab* | Everything that has an "a" in it, followed by anything, followed by a "b", followed by anything.

Search			Columns	
Order ID		Shipped Date		Ship Country
10250		7/12/1996		Brazil
10251		7/15/1996		France
10252		7/11/1996		Belgium
10253		7/16/1996		Brazil
10254		7/23/1996		Switzerland
10255		7/15/1996		Switzerland
10256		7/17/1996		Brazil
10257		7/22/1996		Venezuela
10258		7/23/1996		Austria

LIKE filtering

The **LIKE** filter can process single search patterns using the "%" symbol, retrieving values matching the specified patterns. The following Grid features support LIKE filtering on string-type columns:

- Filter Menu
- Filter Bar with the [filterSettings.showFilterBarOperator](#) property enabled on the Grid [filterSettings](#).
- Custom Filter of Excel filter type.

For example:

Operator | Description

%ab% | Returns all the value that are contains "ab" character.

ab% | Returns all the value that are ends with "ab" character.

%ab | Returns all the value that are starts with "ab" character.

Order ID	Shipped Date	Ship Country
10250	7/12/1996	Brazil
10251	7/15/1996	France
10252	7/11/1996	Belgium
10253	7/16/1996	Brazil
10254	7/23/1996	Switzerland
10255	7/15/1996	Switzerland
10256	7/17/1996	Brazil
10257	7/22/1996	Venezuela
10258	7/23/1996	Austria

By default, the [filterSettings.columns.operator](#) value is **equal**.

Diacritics filter

By default, grid ignores diacritic characters while filtering. To include diacritic characters, set the [filterSettings.ignoreAccent](#) as **true**.

In the following sample, type **aero** in **Name** column to filter diacritic characters.

INDEX.TS

```
import { Grid, Filter, Page } from '@syncfusion/ej2-grids';
import { data } from 'datasource.ts';
Grid.Inject(Page, Filter);
let grid: Grid = new Grid({
  dataSource: data,
  allowFiltering: true,
  filterSettings: { ignoreAccent: true },
```

```

        columns: [
            { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 140 },
            { field: 'Name', headerText: 'Name', width: 170 },
            { field: 'ShipName', headerText: 'Ship Name', width: 170 },
            { field: 'CustomerID', headerText: 'Supplier Name', width: 140 }
        ],
        allowPaging: true,
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Customizing Filter Dialog by using an additional parameter](#)
- [How to implement 'not contains' operator for Grid filtering](#)
- [How to filter custom date ranges in Grid column using date range picker](#)
- [How to filter multiple records using the filter bar template](#)
- [How to change the data source for checkbox filter popup in Grid?](#)
- [How to perform advanced filtering in grid using custom queries](#)

Filter bar in EJ2 JavaScript Grid control

By setting the [allowFiltering](#) to true, the filter bar row will render next to the header, which allows you to filter data. You can filter the records with different expressions depending upon the column type.

Filter bar expressions:

You can enter the following filter expressions (operators) manually in the filter bar.

Expression | Example | Description | Column Type

= | =value | equal | Number

!= | !=value | not equal | Number

|>value | greaterthan | Number

< | <value | lessthan | Number

= | >=value | greaterthanorequal | Number

<= | <=value | lessthanorequal | Number

/value | startswith | String

% | %value | endswith | String

N/A | N/A | **Equal** operator will always be used for date filter. | Date

N/A | N/A | **Equal** operator will always be used for Boolean filter. | Boolean

INDEX.TS

```

import { Grid, Filter } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Filter);
let grid: Grid = new Grid({
    dataSource: data,

```

```

        allowFiltering: true,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
            { field: 'ShipCity', headerText: 'Ship City', width: 100 },
            { field: 'ShipName', headerText: 'Ship Name', width: 100 }
        ],
        height: 273
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
    </style>

```



```

    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter bar template with custom component

The [filterBarTemplate](#) is used to add custom components to a particular column, and does the following functions:

- **create**: Creates custom components.
- **write**: Wires events for custom components.

In the following sample, the dropdown is used as a custom component in the EmployeeID column.

INDEX.TS

```

import { Grid, Filter, Column } from '@syncfusion/ej2-grids';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { data } from './datasource.ts';
Grid.Inject(Filter);
let gridObj: Grid = new Grid({
    dataSource: data,
    allowFiltering: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        {
            field: 'EmployeeID', filterBarTemplate: {
                create: (args: { element: Element, column: Column }) => {
                    const dd = document.createElement('input');
                    dd.id = 'EmployeeID';
                    return dd;
                }
            }
        }
    ]
});

```

```

    },
    write: (args: { element: Element, column: Column }) => {
        let DropDownListObj: DropDownList = new DropDownList({
            dataSource: ['All', '1', '3', '4', '5', '6', '8', '9'],
            fields: { text: 'EmployeeID', value: 'EmployeeID' },
            placeholder: 'Select a value',
            popupHeight: '200px',
            change: function(e) {
                var gridObj =
                (document.getElementsByClassName('e-grid')[0] as any).ej2_instances[0];
                if(e.value == 'All') {

gridObj.removeFilteredColsByField('EmployeeID');
                } else {

gridObj.filterByColumn('EmployeeID', 'equal', parseInt(e.value as any));
                }
            }
        })
        DropDownListObj.appendTo('#EmployeeID');
    },
    textAlign: 'Right', width: 100
},
{ field: 'ShipCity', headerText: 'Ship City', width: 100 },
{ field: 'ShipName', headerText: 'Ship Name', width: 100 }
]);
gridObj.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change default filterbar operator

You can change the filter operator as per the requirement by defining the **filter** property in Grid columns.

In this below demo, we have applied the filter operator **contains** for CustomerID column.

INDEX.TS

```

import { Grid, Page, Filter } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Filter);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowFiltering: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, filter: { operator: "contains" },
headerText: 'Customer ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 220
});

```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

See also

- [How to perform filter by using Wildcard and LIKE operator filter](#)

Filter menu in EJ2 JavaScript Grid control

You can enable filter menu by setting the [filterSettings.type](#) as `Menu`. The filter menu UI will be rendered based on its column type, which allows you to filter data. You can filter the records with different operators.

INDEX.TS

```
import { Grid, Filter } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Filter);
let grid: Grid = new Grid({
  dataSource: data,
  allowFiltering: true,
  filterSettings: { type: 'Menu' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', width: 100 }
  ],
  height: 273
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* [allowFiltering](#) must be set as true to enable filter menu.

* Setting [columns.allowFiltering](#) as false will prevent filter menu rendering for a particular column.

Custom component in filter menu

The [column.filter.ui](#) is used to add custom filter components to a particular column.

To implement custom filter ui, define the following functions:

- **create**: Creates custom component.
- **write**: Wire events for custom component.
- **read**: Read the filter value from custom component.

In the following sample, dropdown is used as custom component in the OrderID column.

INDEX.TS

```
import { Grid, Filter } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { DataManager } from '@syncfusion/ej2-data';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { createElement } from '@syncfusion/ej2-base';
Grid.Inject(Filter);
let grid: Grid = new Grid({
    dataSource: data,
    allowFiltering: true,
    filterSettings: { type: 'Menu' },
    columns: [
        {
            field: 'OrderID', headerText: 'Order ID', width: 120,
            textAlign: 'Right', filter: {
                ui: {
                    create: (args: { target: Element, column: Object })
=> {
                        let db: Object = new DataManager(data);
                        let flValInput: HTMLElement =
createElement('input', { className: 'flm-input' });
                        args.target.appendChild(flValInput);
                        this.dropInstance = new DropDownList({
                            dataSource: new DataManager(data),
                            fields: { text: 'OrderID', value: 'OrderID'
},
                            placeholder: 'Select a value',
                            popupHeight: '200px'
                        });
                        this.dropInstance.appendTo(flValInput);
                    },
                    write: (args: {
                        column: Object, target: Element, parent: any,
                        filteredValue: number | string
                    }) => {
                        this.dropInstance.value = args.filteredValue;
                    },
                    read: (args: { target: Element, column: any,
operator: string, fltrObj: Filter }) => {
                        args.fltrObj.filterByColumn(args.column.field,
args.operator, this.dropInstance.value);
                    }
                }
            }
        },
        { field: 'CustomerID', headerText: 'Customer Name', width: 150
},
    ],
}
```

```

        { field: 'OrderDate', headerText: 'Order Date', width: 130,
format: 'yMd', textAlign: 'Right' },
        { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right'
}
    ]
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
    }
  </style>

```



```

        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing filter menu operators list

You can customize the default filter operator list by defining the [filterSettings.operators](#) property.

The available options are:

- **stringOperator** - Defines the customized string operator list.
- **numberOperator** - Defines the customized number operator list.
- **dateOperator** - Defines the customized date operator list.
- **booleanOperator** - Defines the customized Boolean operator list.

The following sample illustrates customizing the string filter operators.

INDEX.TS

```

import { Grid, Filter } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Filter);
let grid: Grid = new Grid({
    dataSource: data,
    allowFiltering: true,
    filterSettings: { type: 'Menu',
        operators: {
            stringOperator: [
                { value: 'startsWith', text: 'starts with' },
                { value: 'endsWith', text: 'ends with' },
                { value: 'contains', text: 'contains' } ],
        }
    },
    columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {

```

```

        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable different filter for a column

You can use both **Menu** and **CheckBox** filter in a same Grid. To do so, set the [column.filter.type](#) as **Menu** or **CheckBox**.

In the following sample menu filter is enabled by default and checkbox filter is enabled for the CustomerID column using the [column.filter.type](#).

INDEX.TS

```

import { Grid, Filter } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Filter);
let grid: Grid = new Grid({
    dataSource: data,
    allowFiltering: true,
    filterSettings: { type: 'Menu' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 ,
filter: { type : 'CheckBox' } },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter by multiple keywords using filter menu

By default, the filtering action is performed based on the single keyword filter value from the built-in component of the filter menu dialog. Now data grid has an option to perform filtering actions based on multiple keywords instead of a single keyword alone. For this, set [filterSettings.type](#) as `Menu`.

In the following sample, filtering action with multiple keywords can be done by rendering the `MultiSelect` component as custom component in the OrderID column filter menu dialog.

INDEX.TS

```

import { Grid, Filter } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { DataManager } from '@syncfusion/ej2-data';
import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
import { createElement } from '@syncfusion/ej2-base';
Grid.Inject(Filter);
MultiSelect.Inject(CheckBoxSelection);
let grid: Grid = new Grid({
  dataSource: data,
  allowFiltering: true,
  allowPaging: true,
  filterSettings: { type: 'Menu' },
  columns: [
    {
      field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right', filter: {
        ui: {
          create: (args: { target: Element, column: Object }) => {
            let db: Object = new DataManager(data);
            let flValInput: HTMLElement = createElement('input',
{ className: 'flm-input' });
            args.target.appendChild(flValInput);
            this.dropInstance = new MultiSelect({
              dataSource: new DataManager(data),
              fields: { text: 'OrderID', value: 'OrderID' },
              placeholder: 'Select a value',
              popupHeight: '200px',
              allowFiltering: true,
              mode: 'CheckBox',
            });
            this.dropInstance.appendTo(flValInput);
          },
          write: (args: {

```

```

        column: Object, target: Element, parent: any,
        filteredValue: number | string
    }) => {
        let filteredValue = [];
        grid.filterSettings.columns.map((item) => {
            if (item.field === 'OrderID' && item.value) {
                filteredValue.push(item.value);
            }
        });
        if (filteredValue.length > 0) {
            this.dropInstance.value = filteredValue;
        }
    },
    read: (args: { target: Element, column: any, operator:
string, fltrObj: Filter }) => {
        grid.removeFilteredColsByField(args.column.field);
        args.fltrObj.filterByColumn(
            args.column.field,
            'contains',
            this.dropInstance.value
        );
    }
}
    },
    { field: 'CustomerID', headerText: 'Customer Name', width: 150 },
    { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
    { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right' }
]
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add current selection to filter checkbox

By default, the CheckBox filter can only filter the selected items. If filtering is done multiple times on the same column, the previously filtered values in the column will be cleared. Now, it is possible to retain

those previous values by using the **Add current selection to filter** checkbox. This checkbox is displayed when data is searched in the search bar of the CheckBox filter.

The following image describes the above mentioned behavior.

Order ID	Customer Name	Order Date	Freight
102		4/1996 10:10 AM	\$32.38
102		5/1996 12:20 PM	\$11.61
102		3/1996 08:40 AM	\$65.83
102		3/1996 07:50 AM	\$41.34
102		9/1996 12:05 PM	\$51.30
102		0/1996 11:20 AM	\$58.17
102		1/1996 10:00 AM	\$22.98
102		2/1996 10:40 AM	\$148.33
102		5/1996 08:50 PM	\$13.97
10257	Carlos Hernández	7/16/1996 03:20 AM	\$81.91
10258	Roland Mendel	7/17/1996 06:30 PM	\$140.51
10259	Francisco Chang	7/18/1996 01:20 AM	\$3.25

See also

- [How to perform filter by using Wildcard and LIKE operator filter](#)

Excel like filter in EJ2 JavaScript Grid control

You can enable Excel like filter by defining. [filterSettings.type](#) as Excel. The excel menu contains an option such as Sorting, Clear filter, Sub menu for advanced filtering.

INDEX.TS

```
import { Grid, Filter } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Filter);
let grid: Grid = new Grid({
  dataSource: data,
  allowFiltering: true,
  filterSettings: { type: 'Excel' },
  columns: [
```



```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 100 },
        { field: 'ShipName', headerText: 'Ship Name', width: 100 }
    ],
    height: 273
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {

```

```

        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* By default, while opening the excel/checkbox filter in the Grid, the filter dialog will get and display the distinct data from the first 1000 records bound to the Grid to optimize performance. The remaining records will be returned as a result of the search option of the filter dialog.

* However, we can increase the excel/checkbox filter count by modifying the `filterChoiceCount` argument value(as per our need) in the `actionBegin` event when the `requestType` is `filterchoicerequest` or `filtersearchbegin`. This is demonstrated in the below code snippet,

```

`ts
function actionBegin(args: FilterSearchBeginEventArgs) {
if (args.requestType === "filterchoicerequest" || args.requestType === "filtersearchbegin") {
args.filterChoiceCount = 3000;
}
}
`

```

Render checkbox list data in on-demand for excel/checkbox filtering

The Excel/Checkbox filter type of Grid has a restriction where only the first 1000 unique sorted items are accessible to view in the filter dialog checkbox list content by scrolling. This limitation is in place to avoid any rendering delays when opening the filter dialog. However, the searching and filtering processes consider all unique items in that particular column.

The Excel/Checkbox filter in the Grid provides an option to load large data sets on-demand during scrolling to improve scrolling limitation functionality. This is achieved by setting the

[filterSettings.enableInfiniteScrolling](#) property to **true**. This feature proves especially beneficial for managing extensive datasets, enhancing data loading performance in the checkbox list, and allowing interactive checkbox selection with persistence for the selection based on filtering criteria.

The Excel/Checkbox filter retrieves distinct data in ascending order, governed by its [filterSettings.itemsCount](#) property, with a default value of **50**. As the checkbox list data scroller reaches its end, the next dataset is fetched and displayed, with the notable advantage that this process only requests new checkbox list data without redundantly fetching the existing loaded dataset.

Customize the items count for initial rendering

Based on the items count value, the Excel/Checkbox filter gets unique data and displayed in Excel/Checkbox filter content dialog. You can customize the count of on-demand data rendering for Excel/Checkbox filter by adjusting the [filterSettings.itemsCount](#) property. The default value is **50**

`ts

```
grid.filterSettings = { enableInfiniteScrolling = true, itemsCount = 40 };
```

,

It is recommended to keep the itemsCount below **300**. Higher values will result in unwanted whitespace because of DOM maintenance performance degradation.

Customize the loading animation effect

A loading effect is presented to signify that loading is in progress when the checkbox list data scroller reaches the end, and there is a delay in receiving the data response from the server. The loading effect during on-demand data retrieval for Excel/Checkbox filter can be customized using the [filterSettings.loadingIndicator](#) property. The default value is **Shimmer**.

`ts

```
grid.filterSettings = { enableInfiniteScrolling = true, loadingIndicator = 'Spinner' };
```

,

In the provided example, On-Demand Excel filter has been enabled for the EJ2 JavaScript Grid

INDEX.TS

```
import { Grid, Filter, Page, Selection, Sort } from '@syncfusion/ej2-grids';
import { DataManager, Query, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Filter, Page, Selection, Sort);
const urlapi: DataManager = new DataManager({
  url: 'https://services.syncfusion.com/js/production/api/UrlDataSource',
  adaptor: new UrlAdaptor()
});
let grid: Grid = new Grid(
  {
    dataSource: urlapi,
    query: new Query().addParams('dataCount', '10000'),
    allowPaging: true,
    allowFiltering: true,
    allowSorting: true,
    filterSettings: { type: 'Excel', enableInfiniteScrolling: true,
    itemsCount: 40, loadingIndicator: 'Spinner' },
    columns: [
      { field: 'EmployeeID', headerText: 'Employee ID', isPrimaryKey:
true, width: 100 },
```

```

        { field: 'Employees', headerText: 'Employee Name', width: 120 },
        { field: 'Designation', headerText: 'Designation', width: 100 },
        {
            field: 'CurrentSalary', headerText: 'Current Salary',
            format: 'C2',
            textAlign: 'Right', width: 100
        },
    ],
    pageSettings: { pageCount: 5 }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See also

- [How to perform filter by using Wildcard and LIKE operator filter](#)

Searching in EJ2 JavaScript Grid control

You can search records in a Grid, by using the [search](#) method with search key as a parameter. This also provides an option to integrate search text box in grid's toolbar by adding [search](#) item to the [toolbar](#).

To search records, inject the [Search](#) module in the grid.

The clear icon is shown in the Data Grid search text box when it is focused on search text or after typing the single character in the search text box. A single click of the clear icon clears the text in the search box as well as the search results in the Data Grid.

INDEX.TS

```
import { Grid, Search, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Search, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Search'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 150, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 272
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Initial search

To apply search at initial rendering, set the fields, operator, key, and ignoreCase in the [searchSettings](#).

INDEX.TS

```
import { Grid, Search, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Search, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Search'],
  searchSettings: { fields: ['CustomerID'], operator: 'contains', key:
'Ha', ignoreCase: true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 272
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, grid searches all the bound column values. To customize this behavior define the [searchSettings.fields](#) property.

Search operators

The search operator can be defined in the [searchSettings.operator](#) property to configure specific searching.

The following operators are supported in searching:

Operator | Description

startsWith | Checks whether a value begins with the specified value.

endsWith | Checks whether a value ends with the specified value.

contains | Checks whether a value contains the specified value.

equal | Checks whether a value is equal to the specified value.

notEqual | Checks for values not equal to the specified value.

By default, the [searchSettings.operator](#) value is **contains**.

Search by external button

To search grid records from an external button, invoke the [search](#) method.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 260
});
grid.appendTo('#Grid');
let searchBtn: Button = new Button();
searchBtn.appendTo('#search');
document.getElementById('search').addEventListener('click', () => {
    let searchText: string =
(<HTMLInputElement>document.getElementsByClassName('searchtext')[0]).value;
    grid.search(searchText);
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-float-input" style="width: 200px; display: inline-
block;">
            <input type="text" class="searchtext">
            <span class="e-float-line"></span>
            <label class="e-float-text">Search text</label>
        </div>
        <button id="search">Search</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Search specific columns

By default, grid searches all visible columns. You can search specific columns by defining the specific column's field names in the [searchSettings.fields](#) property.

INDEX.TS

```

import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    searchSettings: { fields: ['CustomerID', 'ShipCity', 'ShipName']},

```

```

        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
            { field: 'ShipCity', headerText: 'Ship City', width: 100 },
            { field: 'ShipName', headerText: 'Ship Name', width: 100 }
        ],
        toolbar: ['Search'],
        height: 270
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clear search by external button

To clear the searched grid records from the external button, set [searchSettings.key](#) property as empty string.

INDEX.TS

```

import { Grid, Search, Toolbar } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from './datasource.ts';
Grid.Inject(Search, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Search'],
  searchSettings: { fields: ['CustomerID'], operator: 'contains', key:
'Ha', ignoreCase: true },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 272
});
grid.appendTo('#Grid');
let clearBtn: Button = new Button();
clearBtn.appendTo('#clear');
document.getElementById('clear').addEventListener('click', () => {
  grid.searchSettings.key='';
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="clear">Clear Search</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Search on each key stroke

You can search the Grid data on each key stroke by binding the `keyup` event for the search input element inside the `created` event. Inside the `keyup` handler you can search the Grid by invoking the `search` method of the Grid component.

INDEX.TS

```

import { Grid, Search, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Search, Toolbar);

```

```

let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Search'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 272,
  created: () => {
    document.getElementById(grid.element.id +
"_searchbar").addEventListener('keyup', () => {
      grid.search((event.target as HTMLInputElement).value)
    });
  }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Highlight the search text

You can highlight the search text in the Grid content by adding the style inside the [queryCellInfo](#) event. you can get the search keyword from the [actionBegin](#) event.

INDEX.TS

```

import { Grid, Search, Toolbar, QueryCellInfoEventArgs, SearchEventArgs }
from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Search, Toolbar);
let key: string = '';
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Search'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 272,
    actionBegin: (args: SearchEventArgs) => {
        if (args.requestType === 'searching') {
            key = args.searchString.toLowerCase();
        }
    },
    queryCellInfo: (args: QueryCellInfoEventArgs) => {
        if (key !== '') {
            let cellContent: string = args.data[args.column.field];
            let parsedContent: string = cellContent.toString().toLowerCase();
            if (parsedContent.includes(key.toLowerCase())) {
                let i: number = 0;
                let searchStr: string = '';
                while (i < key.length) {
                    let index: number = parsedContent.indexOf(key[i]);

```

```

        searchStr = searchStr + cellContent.toString()[index];
        i++;
    }
    args.cell.innerHTML = (args.cell as any).innerText.replaceAll(
        searchStr,
        "<span class='customcss'>" + searchStr + "</span>"
    );
}
}
}
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

    <div id="container">
      <div id="Grid"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Perform search operation in Grid using multiple keywords

You can perform a searching operation in the Grid using multiple keywords. This can be achieved by the [actionBegin](#) event of the Grid. In the following sample, we have performed the searching with multiple keywords by using the query property of grid when the requestType is searching in the [actionBegin](#) event.

INDEX.TS

```

import { Grid, Search, Toolbar } from '@syncfusion/ej2-grids';
import { Predicate, Query } from '@syncfusion/ej2-data';
import { data } from './datasource.ts';
Grid.Inject(Search, Toolbar);
let key: string = '';
let values;
let refresh:boolean= false;
let removeQuery:boolean= false;
let valueAssign:boolean= false;
let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Search'],
  columns: [
    {
      field: 'OrderID',headerText: 'Order ID',textAlign:
'Center',width: 120,
    },
    {
      field: 'CustomerID',headerText: 'Customer ID',textAlign:
'Center',width: 150,
    },
    {
      field: 'ShipCity',headerText: 'Ship City',textAlign:
'Center',width: 150,
    },
    {
      field: 'ShipName',headerText: 'Ship Name',textAlign:
'Center',width: 150,
    },
  ],
  searchSettings: {
    fields: ['CustomerID', 'OrderID', 'ShipCity', 'ShipName'],key: ''
  },
  height: 272,
  actionBegin: (args) => {

```

```

        if (args.requestType == 'searching') {
            const keys = args.searchString.split(',');
            var flag = true;
            var predicate;
            if (keys.length > 1) {
                if (grid.searchSettings.key !== '') {
                    values = args.searchString;
                    keys.forEach((key) => {
                        grid.getColumns().forEach((col) => {
                            if (flag) {
                                predicate = new
Predicate(col.field, 'contains', key, true);
                                flag = false;
                            }
                            else {
                                var pre = new
Predicate(col.field, 'contains', key, true);
                                predicate = predicate.or(pre);
                            }
                        });
                    });
                }
                grid.query = new Query().where(predicate);
                grid.searchSettings.key = '';
                refresh = true;
                valueAssign = true;
                removeQuery = true;
                grid.refresh();
            }
        }
    },
    actionComplete: (args) => {
        if (args.requestType === 'refresh' && valueAssign) {
            document.getElementById(grid.element.id + '_searchbar').value =
values;
            valueAssign = false;
        }
        else if (
            document.getElementById(grid.element.id + '_searchbar').value
=== '' &&
            args.requestType === 'refresh' &&
            removeQuery
        ) {
            grid.query = new Query();
            removeQuery = false;
            grid.refresh();
        }
    },
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* Search operation can be performed in foreign key column based on following way.

* When a value is searched on a grid with the foreign key column, a filter query is sent to the foreign key data source, and the appropriate column is filtered depending on the search value. The search query will be sent to the grid data source, and the value of the associated field will be returned.

See Also

- [How to perform searching in Date type column](#)
- [How to search the records in grid on each keystroke](#)
- [How to perform search by using Wildcard and LIKE operator filter](#)

Paging in EJ2 JavaScript Grid control

Paging provides an option to display Grid data in page segments. To enable paging, set the [allowPaging](#) to true. When paging is enabled, pager component renders at the bottom of the grid. Paging options can be configured through the [pageSettings](#).

In the below sample, `pageSize` is calculated based on the grid height by using the `load` event.

To use paging, inject the [Page](#) module in the grid.

INDEX.TS

```
import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 130 },
    { field: 'ShipCity', headerText: 'Ship City', width: 140 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 325,
  load: () => {
    let rowHeight: number = grid.getRowHeight(); //height of the each
row
    let gridHeight: number = grid.height; //grid height
    let pageSize: number = grid.pageSettings.pageSize; //initial page
size
    let pageResize: any = (gridHeight - (pageSize * rowHeight)) /
rowHeight; //new page size is obtained here
    grid.pageSettings.pageSize = pageSize + Math.round(pageResize);
  }
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can achieve better performance by using grid paging to fetch only a pre-defined number of records from the data source.

Template

You can use custom elements inside the pager instead of default elements. The custom elements can be defined by using the [template](#) property. Inside this template, you can access the [CurrentPage](#), [pageSize](#), [pageCount](#), [totalPage](#) and [totalRecordCount](#) values.

INDEX.TS

```

import { Grid, Page, PageEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
Grid.Inject(Page);
let updateTemplate: Function = () => {
    let numeric: NumericTextBox = new NumericTextBox({
        min: 1,
        max: 3,
        step: 1,
        format: '###.##',
        change: (args) => {
            let value: number = args.value;
            grid.goToPage(value);
        }
    });
    numeric.appendTo('#currentPage');
};
let flag: boolean = true;
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    pageSettings: { template: '#template', pageSize: 7 },
    dataBound: () => {
        if (flag) {
            flag = false;
            updateTemplate();
        }
    },
    actionComplete: (args: PageEventArgs) => {
        if (args.requestType === 'paging') {
            updateTemplate();
        }
    }
});

```

```

    }
  });
  grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script id="template" type="text/x-template">
    <div class="e-pagertemplate">
      <div class="col-lg-12 control-section">
        <div class="content-wrapper">
          <input id="currentPage" type="text" value=${currentPage}>
        </div>
      </div>

      <div id="totalPages" class="e-pagertemplatemessage" style="margin-
top:5px;margin-left:30px;border: none; display: inline-block ">
        <span class="e-pagenomsg">${currentPage} of ${totalPages} pages
(${totalRecordsCount} items)</span>
      </div>
    </div>
  </script>

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pager with Page Size Dropdown

The pager Dropdown allows you to change the number of records in the Grid dynamically. It can be enabled by defining the `pageSettings.pageSizes` property as true.

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    pageSettings: { pageSizes: true, pageSize: 8 }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to render Pager at the Top of the Grid

By default, Pager will be rendered at the bottom of the Grid. You can also render the Pager at the top of the Grid by using the `dataBound` event.

INDEX.TS

```

import { Grid, Page, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar);
let initialGridLoad: boolean = true;
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  pageSettings: { pageSizes: true, pageSize: 9 }
});
grid.appendTo('#Grid');
grid.dataBound = () =>{
  if (initialGridLoad) {
    initialGridLoad = false;
    var pager = document.getElementsByClassName('e-gridpager');
    var topElement;
    if (grid.allowGrouping || grid.toolbar) {
      topElement = grid.allowGrouping ?
document.getElementsByClassName('e-groupdroparea') :
document.getElementsByClassName('e-toolbar');
    } else {
      topElement = document.getElementsByClassName('e-gridheader');
    }
    grid.element.insertBefore(pager[0], topElement[0]);
  }
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

During the paging action, the pager component triggers the below three events.

- * The **created** event triggers when Pager is created.
- * The **click** event triggers when the numeric items in the pager is clicked.
- * The **dropDownChanged** event triggers when pageSize DropDownList value is selected.

See Also

- [Group with Paging](#)

Scrolling in EJ2 JavaScript Grid control

The scrollbar will be displayed in the grid when content exceeds the element [width](#) or [height](#). The vertical and horizontal scrollbars will be displayed based on the following criteria:

- The vertical scrollbar appears when the total height of rows present in the grid exceeds its element height.
- The horizontal scrollbar appears when the sum of columns width exceeds the grid element width.
- The [height](#) and [width](#) are used to set the grid height and width, respectively.

The default value for [height](#) and [width](#) is **auto**.

Set width and height

To specify the [width](#) and [height](#) of the scroller in the pixel, set the pixel value to a number.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  height: 315,
  width: 400,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="height:350px;">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Responsive with parent container

Specify the [width](#) and [height](#) as **100%** to make the grid element fill its parent container.

Setting the [height](#) to **100%** requires the grid parent element to have explicit height.

INDEX.TS

```
import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
    { field: 'Freight', headerText: 'Freight', textAlign:
'Right', width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
  ],
  height: '100%',
  allowPaging: true
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-resizable {
        resize: both;
        overflow: auto;
        border: 1px solid red;
        padding: 10px;
        height: 300px;
        min-height: 250px;
        min-width: 250px;
    }
    .e-text{
        font-family: Helvetica, sans-serif;
        font-size: 14px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <p class="e-text"> The parent container can be resizable by dragging the
bottom-right corner.</p>
    <div id="container" class="e-resizable">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sticky Header

You can make the Grid column headers remain fixed while scrolling by using the [enableStickyHeader](#) property.

In the below demo, the Grid headers will be sticky while scrolling the Grid's parent div element.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,

```

```

        enableStickyHeader: true,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ]
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

<div id="container" style="height:350px;">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Scroll to selected row

You can scroll the grid content to the selected row position by using the [rowSelected](#) event.

INDEX.TS

```

import { Grid, RowSelectEventArgs } from '@syncfusion/ej2-grids';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  height: '270',
  width: '100%',
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  rowSelected: rowSelected
});
grid.appendTo('#Grid');
let numeric: NumericTextBox = new NumericTextBox({
  width: 200,
  min: 0,
  showSpinButton: false,
  format: 'N',
  placeholder: 'Enter index to select a row',
  change: onchange
}, '#numeric');
function onchange(): void {
  grid.selectionModule.selectRow(parseInt(numeric.getText(), 10));
}
function rowSelected(args: RowSelectEventArgs) {
  let rowHeight: number =
grid.getRows()[grid.getSelectedRowIndex()[0]].scrollHeight;
  grid.getContent().children[0].scrollTop = rowHeight *
grid.getSelectedRowIndex()[0];
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="height:350px;">
        <input id="numeric" type="text">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hide the scrollbar when the content is not overflown

You can hide the scrollbar of Grid content by using the [hideScroll](#) method when the content doesn't overflow its parent element.

In the following sample, we have invoked the [hideScroll](#) method inside the [dataBound](#) event.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data.slice(0, 5),
  height: '315',
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  dataBound: () => {
    grid.hideScroll();
  }
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="height:350px;">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Frozen in EJ2 JavaScript Grid control

Frozen rows and columns provides an option to make rows and columns always visible in the top and left side of the grid while scrolling.

In this demo, the [frozenColumns](#) is set as '2' and the [frozenRows](#) is set as '3'. Hence, the left two columns and top three rows are frozen.

INDEX.TS

```

import { Grid, Freeze, Selection } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Freeze, Selection);
let grid: Grid = new Grid({
    dataSource: data,
    height: 315,
    allowSelection: false,
    enableHover: false,
    frozenRows: 3,
    frozenColumns: 2,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 },
        { field: 'ShipAddress', headerText: 'Ship Address', width: 170 },

```

```

        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipRegion', headerText: 'Ship Region', width: 150 },
        { field: 'ShipPostalCode', headerText: 'Ship Postal Code', width:
150 },
        { field: 'Freight', headerText: 'Freight', width: 120 }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="height:350px;">
        <div id="Grid"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* Frozen rows and columns should not be set outside the grid view port.

* Frozen Grid will support row and column virtualization feature, which helps to improve the Grid performance while loading a large dataset.

Limitations of Frozen Grid

The following features are not supported in frozen rows and columns:

- Detail Template
- Hierarchy Grid

Freeze particular columns

You can use [isFrozen](#) property to freeze selected columns in grid.

In this demo, the columns with field name **OrderID** and **EmployeeID** is frozen using the **isFrozen** property.

INDEX.TS

```

import { Grid, Freeze, Selection } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Freeze, Selection);
let grid: Grid = new Grid({
    dataSource: data,
    height: 315,
    allowSelection: false,
    enableHover: false,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, isFrozen: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120, isFrozen: true },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 },
        { field: 'ShipAddress', headerText: 'Ship Address', width: 170 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipRegion', headerText: 'Ship Region', width: 150 },
        { field: 'ShipPostalCode', headerText: 'Ship Postal Code', width:
150 },
        { field: 'Freight', headerText: 'Freight', width: 120 }
    ]
});

```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="height:350px;">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

* [isFrozen](#) is not compatible with the Freeze direction feature.

Freeze Direction

You can freeze the Grid columns on the left or right side by using the [column.freeze](#) property and the remaining columns will be movable. The grid will automatically move the columns to the left or right position based on the [column.freeze](#) value.

Types of the [column.freeze](#) directions:

- **Left:** Allows you to freeze the columns at the left.
- **Right:** Allows you to freeze the columns at the right.
- **Fixed:** Allows you to lock the column at a fixed position by ensuring its visibility during horizontal scroll.

In this demo, the **ShipCountry** column is frozen at the left and the **CustomerID** column is frozen at the right side of the content table.

INDEX.TS

```
import { Grid, Freeze } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Freeze);
let grid: Grid = new Grid({
  dataSource: data,
  height: 315,
  enableHover: false,
  frozenRows: 2,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'Freight', headerText: 'Freight', format: 'C2', width: 120
}
    { field: 'CustomerID', headerText: 'Customer ID', width: 150,
freeze: 'Right' },
    { field: 'OrderDate', headerText: 'Order Date', width: 130, format:
'yMd', textAlign: 'Right' },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 },
    { field: 'ShipAddress', headerText: 'Ship Address', width: 170 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150,
freeze: 'Left' }
  ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
```



```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="height:350px;">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* Freeze Direction is not compatible with the [isFrozen](#) and [frozenColumns](#) properties.

Deprecated Methods

Deprecated Methods | [Previous](#) | [Current](#) | [Suggested Alternative Methods](#) | [Example for achieving the same results](#)

getMovableRows() | In the previous architecture of frozen grid, three separate tables were created for the left, right, and movable contents. When calling this method, it would return only the movable table rows (tr's). | In the current architecture, the frozen left, right, and movable sections are applied within a single table. When calling this method, it will return all table rows (tr's) of the entire table. However, in this approach, we have introduced the **e-unfreeze** class for movable cells. This allows us to selectively retrieve the movable rows using the **e-unfreeze** class selector. | **getRows()** | `gridInstance.getMovableRows()[0].querySelectorAll('.e-unfreeze')` // Deprecated

 (or)

 `gridInstance.getRows()[0].querySelectorAll('.e-unfreeze')` // Alternative method

getFrozenRightRows() | In the previous architecture, this method would return only the table rows (tr's) from the freeze right table. | In the current architecture, the frozen left, right, and movable sections are applied within a single table. When calling this method, it will return all the rows (tr's) of the entire table. In this new approach, we have introduced the **e-rightfreeze** class for right freeze cells. As a result, you can now selectively retrieve the right freeze rows using the **e-rightfreeze** class selector. | **getRows()** | `gridInstance.getFrozenRightRows()[0].querySelectorAll('.e-rightfreeze')` // Deprecated

 (or)

 `gridInstance.getRows()[0].querySelectorAll('.e-rightfreeze')` // Alternative method

getMovableRowByIndex()
 getFrozenRowByIndex()
 getFrozenRightRowByIndex() | In the previous architecture, you could select rows by using separate methods for each table section. Like,
 getMovableRowByIndex - select a movable row
 getFrozenRowByIndex - select a freeze row
 getFrozenRightRowByIndex - select a right freeze row. | In the current architecture, the *getMovableRowByIndex*, *getFrozenRightRowByIndex* and *getFrozenRowByIndex* methods all return the same table row (tr) based on the given index. Additionally, class names for table cells (td's) have been separated as follows:
 Left-Freeze : **e-leftfreeze**
 Movable : **e-unfreeze**
 Right-Freeze : **e-rightfreeze**.
 This separation of class names makes it easier to target and customize the cells within the particular row. | **getRowByIndex()** | **To get the left freeze cells:**
 `gridInstance.getRowByIndex(1).querySelectorAll('.e-leftfreeze')`

 To get the movable cells:
 `gridInstance.getRowByIndex(1).querySelectorAll('.e-unfreeze')`

 To get the right freeze cells:
 `gridInstance.getRowByIndex(1).querySelectorAll('.e-rightfreeze')`

getMovableCellFromIndex()
 getFrozenRightCellFromIndex() | *getMovableCellFromIndex()* - select a particular cell in the movable table.
 getFrozenRightCellFromIndex() - select a particular cell in the right freeze table. | In the new approach, you can select a particular cell by using both the *getFrozenRightCellFromIndex* and *getMovableCellFromIndex* methods. | **getCellFromIndex()** | `gridInstance.getCellFromIndex(1,1)`

getMovableDataRows()
 getFrozenRightDataRows()
 getFrozenDataRows() | These methods returns the viewport data rows for the freeze, movable, and right tables separately. | In the new approach, when calling the *getMovableDataRows*, *getFrozenRightDataRows*, and *getFrozenDataRows* methods, returns the entire viewport data rows. You can then select specific cells within these rows using the following selectors
 Left-Freeze : **e-leftfreeze**
 Movable : **e-unfreeze**
 * Right-Freeze : **e-rightfreeze**. | **getDataRows()** | **To get the movable data cells:**
 `gridInstance.getDataRows()[0].querySelectorAll('.e-unfreeze')`

 To get the right freeze data cells:
 `gridInstance.getDataRows()[0].querySelectorAll('.e-rightfreeze')`

 To get the left freeze data cells:
 `gridInstance.getDataRows()[0].querySelectorAll('.e-leftfreeze')`

getMovableColumnHeaderByIndex()
 getFrozenRightColumnHeaderByIndex()
 getFrozenLeftColumnHeaderByIndex() | In the previous architecture, these methods selects the

movable, right freeze, and left freeze headers from the table separately. | In the new approach, when calling the `getMovableColumnHeaderByIndex`, `getFrozenRightColumnHeaderByIndex`, and `getFrozenLeftColumnHeaderByIndex` methods, you will still receive the same results as before. | `getColumnHeaderByIndex()` | `gridInstance.getColumnHeaderByIndex(1)`

When a validation message is displayed in the frozen part (Left, Right, Fixed) of the table, scrolling is prevented until the validation message is cleared.

Virtual scroll in EJ2 JavaScript Grid control

Grid allows you to load large amount of data without performance degradation.

To use virtualization, you need to inject `VirtualScroll` module in grid.

Row Virtualization

Row virtualization allows you to load and render rows only in the content viewport. It is an alternative way of paging in which the data will be loaded while scrolling vertically. To setup the row virtualization, you need to define `enableVirtualization` as true and content height by `height` property.

The number of records displayed in the Grid is determined implicitly by height of the content area. Also, you have an option to define a visible number of records by the `pageSettings.pageSize` property. The loaded data will be cached and reused when it is needed for next time.

INDEX.TS

```
import { Grid, VirtualScroll, Edit, Toolbar } from '@syncfusion/ej2-grids';
Grid.Inject(VirtualScroll, Edit, Toolbar);
let names: string[] = ['TOM', 'Hawk', 'Jon', 'Chandler', 'Monica', 'Rachel',
'Phoebe', 'Gunther', 'Ross', 'Geller', 'Joey', 'Bing', 'Tribbiani',
'Janice', 'Bong', 'Perk', 'Green', 'Ken', 'Adams'];
let hours: number[] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let designation: string[] = ['Manager', 'Engineer 1', 'Engineer 2',
'Developer', 'Tester'];
let status: string[] = ['Completed', 'Open', 'In Progress', 'Review',
'Testing'];
let data: Function = (count: number) => {
    let result: Object[] = [];
    for (let i = 0; i < count; i++) {
        result.push({
            TaskID: i + 1,
            Engineer: names[Math.round(Math.random() * names.length)] ||
names[0],
            Designation: designation[Math.round(Math.random() *
designation.length)] || designation[0],
            Estimation: hours[Math.round(Math.random() * hours.length)] ||
hours[0],
            Status: status[Math.round(Math.random() * status.length)] ||
status[0]
        });
    }
    return result;
};
(<IWindow>window).getStatus = (status: string) => {
    let colors: Object = { 'Completed': 'green', 'Open': 'red', 'In
Progress': '#FB1E77', 'Review': 'brown', 'Testing': '#1EC1FB' };
    return '<span style="color:' + colors[status] + '">' + status +
'</span>';
};
```

```

};
let grid: Grid = new Grid({
  dataSource: data(1000),
  height: 300,
  enableVirtualization: true,
  editSettings: { allowAdding: true, allowEditing: true, allowDeleting:
true, mode: 'Normal' },
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  pageSettings: { pageSize: 50 },
  columns: [
    { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
100, type: 'number', isPrimaryKey: true, validationRules: { required: true }
},
    { field: 'Engineer', width: 100 },
    { field: 'Designation', width: 140, editType: 'dropdownedit',
validationRules: { required: true } },
    { field: 'Estimation', textAlign: 'Right', width: 110, editType:
'numericedit', validationRules: { required: true } },
    { field: 'Status', width: 140, template:
'${getStatus(data.Status)}', editType: 'dropdownedit' }
  ]
});
grid.appendTo('#Grid');
interface IWindow extends Window {
  getStatus?: Function;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column Virtualization

Column virtualization allows you to virtualize columns. It will render columns which are in the viewport. You can scroll horizontally to view more columns.

To setup the column virtualization, set the

[enableVirtualization](#) and

[enableColumnVirtualization](#) properties as `true`.

INDEX.TS

```

import { Grid, VirtualScroll, Edit, Toolbar } from '@syncfusion/ej2-grids';
Grid.Inject(VirtualScroll, Edit, Toolbar);

```

```

let virtualData: Object[] = [];
let names: string[] = ['hardire', 'abramjo01', 'aubucch01', 'Hook',
'Rumpelstiltskin', 'Belle', 'Emma', 'Regina', 'Aurora', 'Elsa', 'Anna',
'Snow White', 'Prince Charming', 'Cora', 'Zelena', 'August', 'Mulan',
'Graham', 'Discord', 'Will', 'Robin Hood', 'Jiminy Cricket', 'Henry',
'Neal', 'Red', 'Aaran', 'Aaren', 'Aarez', 'Aarman', 'Aaron', 'Aaron-James',
'Aarron', 'Aaryan', 'Aaryn', 'Aayan', 'Aazaan', 'Abaan', 'Abbas',
'Abdallah', 'Abdalroof', 'Abdihakim', 'Abdirahman', 'Abdisalam', 'Abdul',
'Abdul-Aziz', 'Abdulbasir', 'Abdulkadir', 'Abdulkarem', 'Abdulkhader',
'Abdullah', 'Abdul-Majeed', 'Abdulmalik', 'Abdul-Rehman', 'Abdur',
'Abdurraheem', 'Abdur-Rahman', 'Abdur-Rehmaan', 'Abel', 'Abhinav',
'Abhisumant', 'Abid', 'Abir', 'Abraham', 'Abu', 'Abubakar', 'Ace', 'Adain',
'Adam', 'Adam-James', 'Addison', 'Addisson', 'Adegbola', 'Adegbolahan',
'Aden', 'Adenn', 'Adie', 'Adil', 'Aditya', 'Adnan', 'Adrian', 'Adrien',
'Aedan', 'Aedin', 'Aedyn', 'Aeron', 'Afonso', 'Ahmad', 'Ahmed', 'Ahmed-
Aziz', 'Ahoua', 'Ahtasham', 'Aiadan', 'Aidan', 'Aiden', 'Aiden-Jack',
'Aiden-Vee'];
function dataSource(): void {
    for (let i: number = 0; i < 1000; i++) {
        virtualData.push({
            'SNo': i + 1,
            'FIELD1': names[Math.floor(Math.random() * names.length)],
            'FIELD2': 1967 + (i % 10), 'FIELD3': Math.floor(Math.random() *
200),
            'FIELD4': Math.floor(Math.random() * 100), 'FIELD5':
Math.floor(Math.random() * 2000), 'FIELD6': Math.floor(Math.random() *
1000), 'FIELD7': Math.floor(Math.random() * 100), 'FIELD8':
Math.floor(Math.random() * 10), 'FIELD9': Math.floor(Math.random() * 10),
'FIELD10': Math.floor(Math.random() * 100), 'FIELD11':
Math.floor(Math.random() * 100), 'FIELD12': Math.floor(Math.random() *
1000), 'FIELD13': Math.floor(Math.random() * 10), 'FIELD14':
Math.floor(Math.random() * 10), 'FIELD15': Math.floor(Math.random() * 1000),
'FIELD16': Math.floor(Math.random() * 200), 'FIELD17':
Math.floor(Math.random() * 300), 'FIELD18': Math.floor(Math.random() * 400),
'FIELD19': Math.floor(Math.random() * 500), 'FIELD20':
Math.floor(Math.random() * 700), 'FIELD21': Math.floor(Math.random() * 800),
'FIELD22': Math.floor(Math.random() * 1000), 'FIELD23':
Math.floor(Math.random() * 2000), 'FIELD24': Math.floor(Math.random() *
150), 'FIELD25': Math.floor(Math.random() * 1000), 'FIELD26':
Math.floor(Math.random() * 100), 'FIELD27': Math.floor(Math.random() * 400),
'FIELD28': Math.floor(Math.random() * 600), 'FIELD29':
Math.floor(Math.random() * 500), 'FIELD30': Math.floor(Math.random() * 300),
        });
    }
}
dataSource();
let grid: Grid = new Grid({
    dataSource: virtualData,
    enableVirtualization: true,
    enableColumnVirtualization: true,
    height: 300,
    editSettings: { allowAdding: true, allowEditing: true, allowDeleting:
true, mode: 'Normal' },
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    columns: [
        { field: 'SNo', headerText: 'S.No', width: 120, isPrimaryKey: true,
validationRules: { required: true } },
    ],
});

```

```

    { field: 'FIELD1', headerText: 'Player Name', width: 140, editType:
'dropdownedit', validationRules: { required: true } },
    { field: 'FIELD2', headerText: 'Year', width: 120, textAlign: 'Right' },
    { field: 'FIELD3', headerText: 'Stint', width: 120, textAlign: 'Right'
},
    { field: 'FIELD4', headerText: 'TMID', width: 120, textAlign: 'Right' },
    { field: 'FIELD5', headerText: 'LGID', width: 120, textAlign: 'Right' },
    { field: 'FIELD6', headerText: 'GP', width: 120, textAlign: 'Right' },
    { field: 'FIELD7', headerText: 'GS', width: 120, textAlign: 'Right' },
    { field: 'FIELD8', headerText: 'Minutes', width: 120, textAlign: 'Right'
},
    { field: 'FIELD9', headerText: 'Points', width: 120, textAlign: 'Right'
},
    { field: 'FIELD10', headerText: 'oRebounds', width: 130, textAlign:
'Right' },
    { field: 'FIELD11', headerText: 'dRebounds', width: 130, textAlign:
'Right' },
    { field: 'FIELD12', headerText: 'Rebounds', width: 120, textAlign:
'Right' },
    { field: 'FIELD13', headerText: 'Assists', width: 120, textAlign:
'Right' },
    { field: 'FIELD14', headerText: 'Steals', width: 120, textAlign: 'Right'
},
    { field: 'FIELD15', headerText: 'Blocks', width: 120, textAlign: 'Right'
},
    { field: 'FIELD16', headerText: 'Turnovers', width: 130, textAlign:
'Right' },
    { field: 'FIELD17', headerText: 'PF', width: 130, textAlign: 'Right' },
    { field: 'FIELD18', headerText: 'fgAttempted', width: 150, textAlign:
'Right' },
    { field: 'FIELD19', headerText: 'fgMade', width: 120, textAlign: 'Right'
},
    { field: 'FIELD20', headerText: 'ftAttempted', width: 150, textAlign:
'Right' },
    { field: 'FIELD21', headerText: 'ftMade', width: 120, textAlign: 'Right'
},
    { field: 'FIELD22', headerText: 'ThreeAttempted', width: 150, textAlign:
'Right' },
    { field: 'FIELD23', headerText: 'ThreeMade', width: 130, textAlign:
'Right' },
    { field: 'FIELD24', headerText: 'PostGP', width: 120, textAlign: 'Right'
},
    { field: 'FIELD25', headerText: 'PostGS', width: 120, textAlign: 'Right'
},
    { field: 'FIELD26', headerText: 'PostMinutes', width: 120, textAlign:
'Right' },
    { field: 'FIELD27', headerText: 'PostPoints', width: 130, textAlign:
'Right' },
    { field: 'FIELD28', headerText: 'PostoRebounds', width: 130, textAlign:
'Right' },
    { field: 'FIELD29', headerText: 'PostdRebounds', width: 130, textAlign:
'Right' },
    { field: 'FIELD30', headerText: 'PostRebounds', width: 130, textAlign:
'Right', editType: 'numericedit', validationRules: { required: true } }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column's [width](#) is required for column virtualization. If column's width is not defined then Grid will consider its value as **200px**.

Virtualization with Grouping

Both the row and column virtualization can be used along with grouping. At initial rendering, the virtual height of scrollbar will be set based on the total number of records and after grouping, it will be refreshed based on the grouped state(expand/collapse). While collapse the group caption row in current viewport then the next view page grouped records are shown.

The collapsed/expanded state will persist only for local dataSource while scrolling.

Limitations for virtual scrolling

- While using column virtual scrolling, column width should be in the pixel. Percentage values are not accepted.
- Due to the element height limitation in browsers, the maximum number of records loaded by the grid is limited by the browser capability.
- The cell selection is not supported for both row and column virtual scrolling.
- Virtual scrolling is not compatible with batch editing, detail template, rowspan, colspan and hierarchy features.
- Group expand and collapse state will not be persisted.
- Since data is virtualized in grid, the aggregated information and total group items are displayed based on the current view items. To get these information regardless of the view items, refer to the

[Group with Page](#) topic.

- The page size provided must be two times larger than the number of visible rows in the grid. If the page size is failed to meet this condition then the size will be determined by grid.
- The height of the grid content is calculated using the row height and total number of records in the data source and hence features which changes row height such as text wrapping are not supported. If you want to increase the row height to accommodate the content then you can specify the row height as below to ensure all the table rows are in same height.

```

.e-grid .e-row {
height: 2em;
}

```

- Programmatic selection using the [selectRows](#) method is not supported in virtual scrolling.

Browser height limitation in virtual scrolling and solution

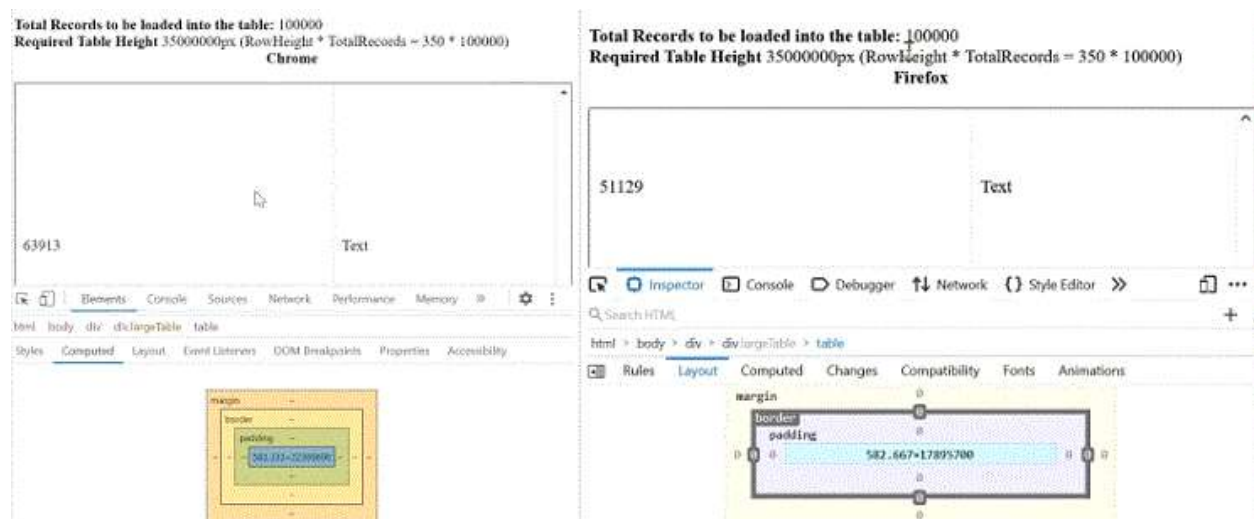
You can load millions of records in the Grid by using virtual scrolling, where the grid loads and renders rows on-demand while scrolling vertically. As a result, Grid lightens the browser's load by minimizing the DOM elements and rendering elements visible in the viewport. The height of the grid is calculated using the Total Records Count * [Row Height](#) property.

The browser has some maximum pixel height limitations for the scroll bar element. The content placed above the maximum height can't be scrolled if the element height is greater than the browser's maximum height limit. The browser height limit affects the virtual scrolling of the grid. When a large number of records are bound to the Grid, it can only display the records until the maximum height limit of the browser. Once the browser's height limit is reached while scrolling, the user won't be able to scroll further to view the remaining records.

For example, if the row height is set as 30px and the total record count is 1000000(1 million), then the height of the grid element will be 30,000,000 pixels. In this case, the browser's maximum height limit for a div is about 22,369,600 (The maximum pixel height limitation differs for different browsers). The records above the maximum height limit of the browser can't be scrolled.

This height limitation is not related to the Grid component. It fully depends on the default behavior of the browser. The same issue is reproduced in the normal HTML table too.

The following image illustrates the height limitation issue of a normal HTML table in different browsers (Chrome and Firefox).



Grid component also faced the same issue as mentioned in the below image.



The Grid has an option to overcome this limitation of the browser in the following ways.

Solution 1: Using external buttons

You can prevent the height limitation problem in the browser when scrolling through millions of records by loading the segment of data through different strategy.

In the following sample, Grid is rendered with a large number of records(nearly 2 million). Here, you can scroll 0.5 million records at a time in Grid. Once you reach the last page of 0.5 million records, the **Load Next Set** button will be shown at the bottom of the Grid. By clicking that button, you can view the next set of 0.5 million records in Grid. Also, the **Load Previous Set** button will be shown at the top of the Grid to load the previous set of 0.5 million records.

Let's see the step by step procedure for how we can overcome the limitation in the Syncfusion Grid component.

1. Create a custom adaptor by extending UrlAdaptor and binding it to the grid DataSource property. In the processQuery method of the custom adaptor, we handled the Skip query based on the current page set to perform the data operation with whole records on the server.

```
`ts
```

```
class CustomUrlAdaptor extends UrlAdaptor {
  processQuery(args) {
    if (arguments[1].queries) {
      for (var i = 0; i < arguments[1].queries.length; i++) {
        if (arguments[1].queries[i].fn === 'onPage') {
          // pageSet - defines the number of segments that we are going to split the 2million records. In this
          // example we have considered 0.5 million records for each set so the pageSet is 1, 2, 3 and 4.
          // maxRecordsPerPageSet – In this example we define the value as 0.5 million.
          // gridPageSize – the pageSize that we have defined in the Grid pageSettings->pageSize property
          // customize the pageIndex based on the current pageSet (It send the skip query including the previous
          // pageSet ) so that the other operations performed for total 2millions records instead of 0.5 million alone.
```

```

arguments[1].queries[i].e.pageIndex = (((pageSet - 1) * maxRecordsPerPageSet) / gridPageSize) +
arguments[1].queries[i].e.pageIndex;
}
}
}
let original = super.processQuery.apply(this, arguments);
return original;
}
}
let data: DataManager = new DataManager({
adaptor: new CustomUrlAdaptor,
url: "Home/UrlDatasource"
});
`

```

2. Render the grid by define the following features.

```

`ts
let grid: Grid = new Grid({
dataSource: data,
enableVirtualization: true,
pageSettings: {pageSize: 50},
height: 360,
beforeDataBound: beforeDataBound,
columns: [
{ field: 'OrderID', width: 120, headerText: 'Order ID', textAlign: 'Right' }
.....
.....
]
});
`

```

3. In the beforeDataBound event, we set the args.count as 0.5 million to perform scrolling with 0.5 million records and all the data operations are performed with whole records which is handled using the custom adaptor. And also particular segment records count is less than 0.5 million means it will directly assigned the original segmented count instead of 0.5 million.

```

`ts
beforeDataBound(args) {
// storing the total records count which means 2 million records count
totalRecords = args.count;
// change the count with respect to maxRecordsPerPageSet (maxRecordsPerPageSet = 500000)
args.count = args.count - ((pageSet - 1) maxRecordsPerPageSet) > maxRecordsPerPageSet
?maxRecordsPerPageSet : args.count - ((pageSet - 1) maxRecordsPerPageSet);
}
`

```

4. Render “Load Next Set” button and “Load Previous Set” button at bottom and top of the grid component.

```

`ts
let button: Button = new Button({
cssClass: 'e-info prevbtn',
onClick: prevBtnClick,
content: 'Load Previous Set...'
});
let grid: Grid = new Grid({
dataSource: data,
enableVirtualization: true,
pageSettings: {pageSize: 50},
height: 360,
beforeDataBound: beforeDataBound,
columns: [
{ field: 'OrderID', width: 120, headerText: 'Order ID', textAlign: 'Right' }
.....
.....
]
});
let button: Button = new Button({
cssClass: 'e-info nextbtn',
onClick: nextBtnClick,
content: 'Load Next Set...'
}

```

```
});  
,
```

5. While click on the **Load Next Set / Load Previous Set** button corresponding page data set is loaded to view remaining records of total 2 millions records after doing some simple calculation.

```
`ts  
  
// Triggered when clicking the Previous/ Next button.  
prevNxtBtnClick(args) {  
  if (grid.element.querySelector('.e-content') && grid.element.querySelector('.e-  
content').getAttribute('aria-busy') === 'false') {  
    // Increase/decrease the pageSet based on the target element.  
    pageSet = args.target.classList.contains('prevbtn') ? --pageSet : ++pageSet;  
    this.rerenderGrid(); // Re-render the Grid component.  
  }  
}  
,
```

You can view the hosted link for this sample [here](#).

LOAD PREVIOUS SET...				
Order ID	Customer ID	Freight	Country	Status
1	Alfki	\$12.30	📍Denmark	Active
2	Anatr	\$3.30	📍Brazil	Inactive
3	Anton	\$4.30	📍Germany	Active
4	Blomp	\$5.30	📍Austria	Inactive
5	Bolid	\$6.30	📍Switzerland	Active
6	Hanar	\$12.30	📍France	Active
7	Thomas	\$3.30	📍Itali	Inactive
8	Jack	\$4.30	📍Austria	Active
9	Alfki	\$5.30	📍USA	Inactive
10	Hanar	\$6.30	📍Belgium	Active
11	Alfki	\$3.30	📍Denmark	Active
LOAD NEXT SET...				

If you perform grid actions such as filtering, sorting, etc., after scrolling through the 0.5 million data, the Grid performs those data actions with the whole records, not just the current loaded 0.5 million data.

Solution 2: Using RowHeight property

You can reduce the [row height](#) using the [rowHeight](#) property of the Grid. It will reduce the overall height to accommodate more rows. But this approach optimizes the limitation, but if the height limit is reached after reducing row height also, you have to opt for the previous solution or use paging.

In the following image, you can see how many records will be scrollable when setting rowHeight to "36px" and "30px".

RowHeight = 36px

Order ID	Customer ID	Weight	Country	Status
1	Alfo	\$12.30	Denmark	Active
2	Amir	\$3.30	Brazil	Inactive
3	Anton	\$4.30	Germany	Active
4	Bomp	\$5.30	Austria	Inactive
5	Bold	\$6.30	Switzerland	Active
6	Hansr	\$12.30	France	Active
7	Thomas	\$3.30	Italy	Inactive
8	Jack	\$4.30	Austria	Active

RowHeight = 30px

Order ID	Customer ID	Weight	Country	Status
1	Alfo	\$12.30	Denmark	Active
2	Amir	\$3.30	Brazil	Inactive
3	Anton	\$4.30	Germany	Active
4	Bomp	\$5.30	Austria	Inactive
5	Bold	\$6.30	Switzerland	Active
6	Hansr	\$12.30	France	Active
7	Thomas	\$3.30	Italy	Inactive
8	Jack	\$4.30	Austria	Active
9	Alfo	\$5.30	USA	Inactive
10	Hansr	\$6.30	Belgium	Active

Solution 3: Using paging instead of virtual scrolling

Similar to virtual scrolling, the [paging](#) feature also loads the data in an on-demand concept. Pagination is also compatible with all the other features (Grouping, Editing, etc.) in Grid. So, use the paging feature instead of virtual scrolling to view a large number of records in the Grid without any kind of performance degradation or browser height limitation.

Infinite scroll in EJ2 JavaScript Grid control

Infinite scrolling is used to load a huge amount of data without degrading the Grid performance. This feature works like the lazy loading concept, which means the buffer data is loaded only when the scrollbar reaches the end of the scroller.

To enable Infinite scrolling, set `enableInfiniteScrolling` property as true.

* In this feature, Grid will not make a new data request when you visit the same page again.

INDEX.TS

```
import { Grid, InfiniteScroll } from '@syncfusion/ej2-grids';
Grid.Inject(InfiniteScroll);
```



```

let names: string[] = ['TOM', 'Hawk', 'Jon', 'Chandler', 'Monica', 'Rachel',
'Phoebe', 'Gunther', 'Ross', 'Geller', 'Joey', 'Bing', 'Tribbiani',
'Janice', 'Bong', 'Perk', 'Green', 'Ken', 'Adams'];
let hours: number[] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let designation: string[] = ['Manager', 'Engineer 1', 'Engineer 2',
'Developer', 'Tester'];
let status: string[] = ['Completed', 'Open', 'In Progress', 'Review',
'Testing'];
let data: Function = (count: number) => {
    let result: Object[] = [];
    for (let i = 0; i < count; i++) {
        result.push({
            TaskID: i + 1,
            Engineer: names[Math.round(Math.random() * names.length)] ||
names[0],
            Designation: designation[Math.round(Math.random() *
designation.length)] || designation[0],
            Estimation: hours[Math.round(Math.random() * hours.length)] ||
hours[0],
            Status: status[Math.round(Math.random() * status.length)] ||
status[0]
        });
    }
    return result;
};
(<IWindow>window).getStatus = (status: string) => {
    let colors: Object = { 'Completed': 'green', 'Open': 'red', 'In
Progress': '#FB1E77', 'Review': 'brown', 'Testing': '#1EC1FB' };
    return '<span style="color:' + colors[status] + '">' + status +
'</span>';
};
let grid: Grid = new Grid({
    dataSource: data(1000),
    height: 300,
    enableInfiniteScrolling: true,
    pageSettings: { pageSize: 50 },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
50, type: 'number' },
        { field: 'Engineer', width: 100 },
        { field: 'Designation', width: 100 },
        { field: 'Estimation', textAlign: 'Right', width: 100 },
        { field: 'Status', width: 100, template: '${getStatus(data.Status)}'
    }
    ]
});
grid.appendTo('#Grid');
interface IWindow extends Window {
    getStatus?: Function;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
<meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

InitialBlocks

You can define the initial loading pages count by using `infiniteScrollSettings.initialBlocks` property. By default, this feature loads three pages in initial rendering.

In the below demo, we have changed this property value to load five page records instead of three.

INDEX.TS

```

import { Grid, InfiniteScroll } from '@syncfusion/ej2-grids';
Grid.Inject(InfiniteScroll);
let names: string[] = ['TOM', 'Hawk', 'Jon', 'Chandler', 'Monica', 'Rachel',
    'Phoebe', 'Gunther', 'Ross', 'Geller', 'Joey', 'Bing', 'Tribbiani',
    'Janice', 'Bong', 'Perk', 'Green', 'Ken', 'Adams'];
let hours: number[] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let designation: string[] = ['Manager', 'Engineer 1', 'Engineer 2',
    'Developer', 'Tester'];
let status: string[] = ['Completed', 'Open', 'In Progress', 'Review',
    'Testing'];
let data: Function = (count: number) => {
    let result: Object[] = [];
    for (let i = 0; i < count; i++) {
        result.push({
            TaskID: i + 1,
            Engineer: names[Math.round(Math.random() * names.length)] ||
names[0],
            Designation: designation[Math.round(Math.random() *
designation.length)] || designation[0],
            Estimation: hours[Math.round(Math.random() * hours.length)] ||
hours[0],
            Status: status[Math.round(Math.random() * status.length)] ||
status[0]
        });
    }
    return result;
};
(<IWindow>window).getStatus = (status: string) => {
    let colors: Object = { 'Completed': 'green', 'Open': 'red', 'In
Progress': '#FB1E77', 'Review': 'brown', 'Testing': '#1EC1FB' };
    return '<span style="color:' + colors[status] + '>' + status +
'</span>';
};
let grid: Grid = new Grid({
    dataSource: data(1000),
    height: 300,
    enableInfiniteScrolling: true,
    infiniteScrollSettings: { initialBlocks: 5 },

```

```

    pageSettings: { pageSize: 50 },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
50, type: 'number' },
        { field: 'Engineer', width: 100 },
        { field: 'Designation', width: 100 },
        { field: 'Estimation', textAlign: 'Right', width: 100 },
        { field: 'Status', width: 100, template: '${getStatus(data.Status)}'
    }
    ]
});
grid.appendTo('#Grid');
interface IWindow extends Window {
    getStatus?: Function;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
    
```

```

        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cache Mode

Cache is used to store the loaded rows object in the Grid instance which can be reused for creating the row elements whenever you scroll to already visited page. Also, this mode maintains row elements based on the `infiniteScrollSettings.maxBlocks` count value, once this limit exceeds then it will remove row elements from DOM for new rows.

To enable the cache mode in Infinite scrolling, set `infiniteScrollSettings.enableCache` property as true.

INDEX.TS

```

import { Grid, InfiniteScroll } from '@syncfusion/ej2-grids';
Grid.Inject(InfiniteScroll);
let names: string[] = ['TOM', 'Hawk', 'Jon', 'Chandler', 'Monica', 'Rachel',
'Phoebe', 'Gunther', 'Ross', 'Geller', 'Joey', 'Bing', 'Tribbiani',
'Janice', 'Bong', 'Perk', 'Green', 'Ken', 'Adams'];
let hours: number[] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let designation: string[] = ['Manager', 'Engineer 1', 'Engineer 2',
'Developer', 'Tester'];
let status: string[] = ['Completed', 'Open', 'In Progress', 'Review',
'Testing'];
let data: Function = (count: number) => {
    let result: Object[] = [];
    for (let i = 0; i < count; i++) {
        result.push({

```

```

        TaskID: i + 1,
        Engineer: names[Math.round(Math.random() * names.length)] ||
names[0],
        Designation: designation[Math.round(Math.random() *
designation.length)] || designation[0],
        Estimation: hours[Math.round(Math.random() * hours.length)] ||
hours[0],
        Status: status[Math.round(Math.random() * status.length)] ||
status[0]
    });
    }
    return result;
};
(<IWindow>window).getStatus = (status: string) => {
    let colors: Object = { 'Completed': 'green', 'Open': 'red', 'In
Progress': '#FB1E77', 'Review': 'brown', 'Testing': '#1EC1FB' };
    return '<span style="color:' + colors[status] + '">' + status +
'</span>';
};
let grid: Grid = new Grid({
    dataSource: data(1000),
    height: 300,
    enableInfiniteScrolling: true,
    infiniteScrollSettings: { enableCache: true },
    pageSettings: { pageSize: 50 },
    columns: [
        { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
50, type: 'number' },
        { field: 'Engineer', width: 100 },
        { field: 'Designation', width: 100 },
        { field: 'Estimation', textAlign: 'Right', width: 100 },
        { field: 'Status', width: 100, template: '${getStatus(data.Status)}'
    }
    ]
});
grid.appendTo('#Grid');
interface IWindow extends Window {
    getStatus?: Function;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations for Infinite Scrolling

- Due to the element height limitation in browsers, the maximum number of records loaded by the grid is limited due to the browser capability.
- Initial loading rows total height must be greater than the viewport height.
- Cell selection will not be persisted in cache mode.
- Infinite scrolling is not compatible with batch editing, detail template and hierarchy features.
- The group records cannot be collapsed in cache mode.
- The aggregated information and total group items are displayed based on the current view items. To get these information regardless of the view items, refer to the

[Group with Page](#) topic.

- Programmatic selection using the [selectRows](#) and [selectRow](#) method is not supported in infinite scrolling.

Selection

Selection in EJ2 JavaScript Grid control

Selection provides an option to highlight a row or a cell or a column. It can be done through simple mouse down or arrow keys. To disable selection in the Grid, set the [allowSelection](#) to false.

The grid supports two types of selection that can be set by using the [selectionSettings.type](#). They are:

- **Single:** The **Single** value is set by default, and it only allows selection of a single row or a cell or a column.
- **Multiple:** Allows you to select multiple rows or cells or columns.

To perform the multi-selection, press and hold CTRL key and click the desired rows or cells or columns. To select range of rows or cells or columns, press and hold the SHIFT key and click the rows or cells or columns.

INDEX.TS

```
import { Grid, Selection } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  selectionSettings: { type: 'Multiple' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection mode

The grid supports three types of selection mode that can be set by using the [selectionSettings.mode](#). They are:

- **Row:** The **Row** value is set by default, and allows you to select only rows.
- **Cell:** Allows you to select only cells.
- **Both:** Allows you to select rows and cells at the same time.

INDEX.TS

```

import { Grid, Selection } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    selectionSettings: { mode: 'Both' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Touch interaction

When you tap a grid row on touchscreen device, the tapped row is selected. It also shows a popup

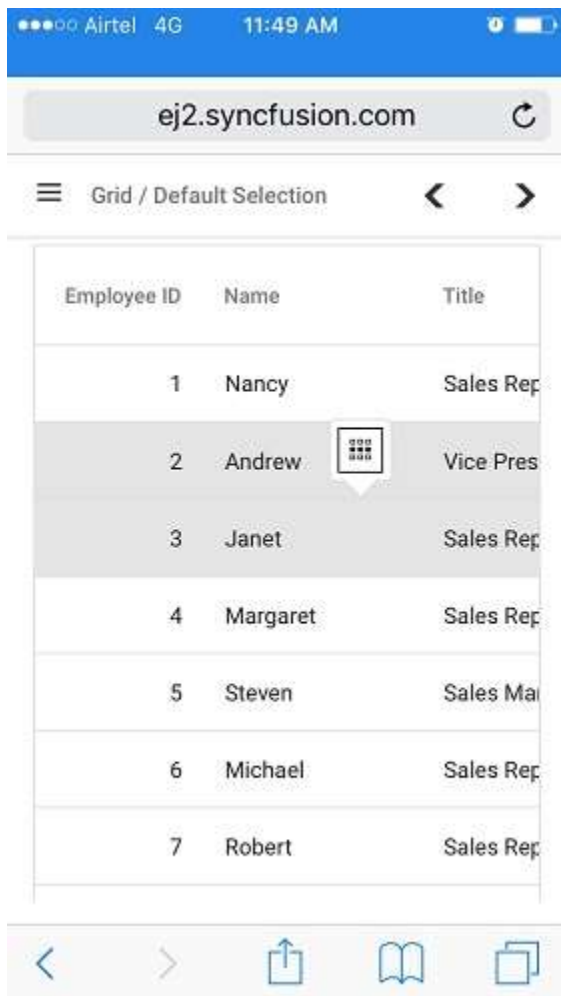


for multi-row selection. To select multiple rows or cells, tap the popup and then tap the desired rows or cells.



Multi-selection requires the selection [type](#) to be **multiple**.

The following screenshot represents a grid touch selection in the device.



Row selection in EJ2 JavaScript Grid control

Select row at initial rendering

To select a row at initial rendering, set the [selectedRowIndex](#) value.

INDEX.TS

```
import { Grid, Selection } from '@syncfusion/ej2-grids';
```

```
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  selectionSettings: { type: 'Multiple', mode: 'Both' },
  selectedRowIndex: 1,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
```

```

        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Get selected row indexes

You can get the selected row indexes by using the [getSelectedRowIndexes](#) method.

INDEX.TS

```

import { Grid, RowSelectEventArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    selectionSettings: {type: 'Multiple'},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315,
    rowSelected: rowSelected
});
grid.appendTo('#Grid');
function rowSelected(args: RowSelectEventArgs) {
    let selectedrowindex: number[] = grid.getSelectedRowIndexes(); // get
the selected row indexes.
}

```

```

    alert(selectedrowindex); // to alert the selected row indexes.
    let selectedrecords: Object[] = grid.getSelectedRecords(); // get the
    selected records.
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Simple multiple row selection

You can select multiple rows by clicking on rows one by one. This will not deselect the previously selected rows. To deselect the previously selected row, you can click on the selected row. You can enable this behavior by using [selectionSettings.enableSimpleMultiRowSelection](#) property.

INDEX.TS

```

import { Grid, Selection } from '@syncfusion/ej2-grids';
import { sdata } from './datasource.ts';
Grid.Inject(Selection);
let grid: Grid = new Grid({
    dataSource: sdata,
    selectionSettings: {type: 'Multiple', enableSimpleMultiRowSelection:
true},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Toggle selection

The Toggle selection allows to perform selection and unselection of the particular row or cell or column. To enable toggle selection, set [enableToggle](#) property of the selectionSettings as true. If you click on the selected row or cell or column then it will be unselected and vice versa.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    selectionSettings: {enableToggle: true},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="Grid"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* If multi selection is enabled, then first click on any selected row (without pressing Ctrl key), it will clear the multi selection and in second click on the same row, it will be unselected.

Clear selection programmatically

You can clear the Grid selection programmatically by using the [clearSelection](#) method.

In the demo below, we initially selected the third row using [selectedRowIndex](#). You can clear this selection by calling the [clearSelection](#) method in the button click event.

INDEX.TS

```

import { Grid, Selection, Page } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from './datasource.ts';
Grid.Inject(Page, Selection);
let grid: Grid = new Grid({
  dataSource: data,
  allowSelection: true,
  allowPaging: true,
  selectionSettings: { type: 'Multiple' },
  selectedRowIndex: 2,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 130 },
    { field: 'Freight', headerText: 'Freight', format: 'C2', width: 100
}
  ],
  pageSettings: { pageSizes: true, pageSize: 5 }
});
grid.appendTo('#Grid');
let show: Button = new Button({ cssClass: 'e-flat' }, '#show');
document.getElementById('show').onclick = () => {
  let grid = document.getElementById('Grid').ej2_instances[0];
  grid.clearSelection();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="show">Clear Selection</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Get selected records on various pages

Enabling the [selectionSettings.persistSelection](#) property will persist the selection in all Grid operations.

So the selection will be maintained on every page even after navigating to another page.

You can get the selected records using the [getSelectedRecords](#) method.

INDEX.TS

```
import { Grid, Selection, RowSelectEventArgs, Page } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from './datasource.ts';
Grid.Inject(Page, Selection);
let grid: Grid = new Grid({
    dataSource: data,
    allowSelection: true,
    allowPaging: true,
    selectionSettings: { type: 'Multiple', persistSelection: true },
    columns: [
        { type: 'checkbox', width: 50 },
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
isPrimaryKey: true, width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 130 },
        { field: 'Freight', headerText: 'Freight', format: 'C2', width: 100
    ]
},
    pageSettings: { pageSizes: true, pageSize: 5 }
});
grid.appendTo('#Grid');
let show: Button = new Button({ cssClass: 'e-flat' }, '#show');
document.getElementById('show').onclick = () => {
    let grid = document.getElementById('Grid').ej2_instances[0];
    let selectedrecords: Object[] = grid.getSelectedRecords();
    let selectedRecordsCount: number = selectedrecords.length;
    alert(selectedRecordsCount);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <script id="template" type="text/x-template">
        <input id='${OrderID}' value='${Freight}' class='custemp'
type='text' style='width: 100%'>
    </script>
    <div id="container">
        <button id="show">Selected Records</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* To persist the grid selection, it is necessary to define any one of the columns as a primary key using the [columns.isPrimaryKey](#) property.

Cell selection in EJ2 JavaScript Grid control

Cell selection can be done through simple mouse down or arrow keys (up, down, left, and right).

The grid supports two types of cell selection mode that can be set by using the [selectionSettings.cellSelectionMode](#). They are:

- **Flow:** The **Flow** value is set by default. The range of cells are selected between the start index and end index that includes in between cells of rows.
- **Box:** Range of cells are selected from the start and end column indexes that includes in between cells of rows within the range.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  enableHover: false,
  selectionSettings: { cellSelectionMode: 'Box', type: 'Multiple', mode:
'Cell' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell selection requires the [selectionSettings.mode](#) to be **Cell** or **Both**, and [type](#) should be **Multiple**.

Toggle selection

The Toggle selection allows to perform selection and unselection of the particular row or cell or column. To enable toggle selection, set [enableToggle](#) property of the selectionSettings as true. If you click on the selected row or cell or column then it will be unselected and vice versa.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';

```



```

let grid: Grid = new Grid({
    dataSource: data,
    selectionSettings: {enableToggle: true},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
    </style>

```

```

        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
    </head>
    <body>

        <div id="container">
            <div id="Grid"></div>
        </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
    </body></html>

```

* If multi selection is enabled, then first click on any selected row (without pressing Ctrl key), it will clear the multi selection and in second click on the same row, it will be unselected.

Column selection in EJ2 JavaScript Grid control

Column selection can be done through simple mouse down or arrow keys.

You can enable column selection by setting the [selectionSettings.allowColumnSelection](#) property as true.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    enableHover: false,
    selectionSettings: { allowColumnSelection: true, type: 'Multiple' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});

```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

Check box selection in EJ2 JavaScript Grid control

Checkbox selection provides an option to select multiple grid records with help of checkbox in each row.

To render the checkbox in each grid row, you need to use checkbox column with type as `checkbox` using the column `type` property.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
    dataSource: data,
    columns: [
        { type: 'checkbox' },
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 315
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* By default, selection is allowed by clicking a grid row or checkbox in that row. To allow selection only through checkbox, you can set the

[selectionSettings.checkboxOnly](#) property to true.

* Selection can be persisted in all the operations using the [selectionSettings.persistSelection](#) property. For persisting selection on the grid, any one of the columns should be defined as a primary key using the [columns.isPrimaryKey](#) property.

Checkbox selection mode

In checkbox selection, selection can also be done by clicking on rows. This selection provides two types of Checkbox Selection mode which can be set by using the following API, [selectionSettings.checkboxMode](#). The modes are;

- **Default:** This is the default value of the checkboxMode. In this mode, user can select multiple rows by clicking rows one by one.
- **ResetOnRowClick:** In ResetOnRowClick mode, when user clicks on a row it will reset previously selected row. Also you can perform multiple-selection in this mode by press

and hold CTRL key and click the desired rows. To select range of rows, press and hold the SHIFT key and click the rows.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  selectionSettings: {checkboxMode: 'ResetOnRowClick'},
  columns: [
    { type: 'checkbox', width: 50 },
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent specific rows from being selected in checkbox selection

You can prevent specific rows from being selected in the checkbox selection mode by hiding the checkboxes using the [rowDataBound](#) event. You achieve this by setting the [isSelectable](#) argument as false in the [rowDataBound](#) event based on certain conditions as per the needs of the application.

In the following sample, the selection of specific rows has been prevented based on the `isSelectable` argument in the `rowDataBound` event.

INDEX.TS

```
import { Grid, Edit, Page, Filter, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Page, Filter, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    selectionSettings: { persistSelection: true },
    allowFiltering: true,
    filterSettings: { type: 'CheckBox' },
    pageSettings: { pageSize: 20 },
    editSettings: {
        allowEditing: true,
        allowAdding: true,
        allowDeleting: true,
        mode: 'Normal',
    },
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel', 'Search'],
    rowDataBound: rowDataBound,
    columns: [
        { type: 'checkbox', width: 120 },
        { field: 'List', headerText: 'List', width: 120 },
        { field: 'OrderID', isPrimaryKey: true, headerText: 'Order ID',
width: 150 },
        { field: 'CustomerID', headerText: 'CustomerID', width: 150 },
        { field: 'EmployeeID', headerText: 'Employee ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 }
    ]
});
grid.appendTo('#Grid');
for (let i = 0; i < data.length; i++) {
    data[i]['List'] = i + 1;
}
function rowDataBound(args): void {
    args.isSelectable = args.data.List % 5 === 0;
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Aggregates

Aggregates in EJ2 JavaScript Grid control

Aggregate values are displayed in the footer, group footer, or group caption of the Grid. It can be configured through `aggregates` property.

`Field` and `type` are the minimum properties required to represent an aggregate column.

To use the aggregate feature, you have to inject the `Aggregate` module.

By default, the aggregate value can be displayed in the footer, group, and caption cells. To show the aggregate value in one of the cells, use the `footerTemplate`, `groupFooterTemplate`, or `groupCaptionTemplate` property.

Built-in aggregate types

The aggregate type should be specified in the `type` property to configure an aggregate column.

The built-in aggregates are,

- Sum
- Average
- Min
- Max
- Count
- Truecount
- Falsecount

* Multiple aggregates can be used for an aggregate column by setting the `type` property with an array of aggregate types.

* Multiple types for a column is supported only when one of the aggregate templates is used.

Footer aggregate in EJ2 JavaScript Grid control

Footer aggregate value is calculated for all the rows, and it is displayed in the footer cells. Use the `footerTemplate` property to render the aggregate value in footer cells.

INDEX.TS

```
import { Grid, Aggregate } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Aggregate);
let grid: Grid = new Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'Freight', headerText: 'Freight', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 210,
  aggregates: [{
    columns: [{
```

```

        type: 'Sum',
        field: 'Freight',
        footerTemplate: 'Sum: ${Sum}'
    }
  ],
  {
    columns: [{
      type: 'Max',
      field: 'Freight',
      footerTemplate: 'Max: ${Max}'
    }]
  }
]);
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <div id="toolbar-template">
      <div id="refresh" title="Refresh">
        <button class="e-btn e-flat">
          <span class="e-btn-icon e-icons e-refresh"></span>
        </button>
      </div>
    </div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The aggregate values must be accessed inside the template using their corresponding [type](#) name.

How to format aggregate value

You can format the aggregate value result by using the [format](#) property.

INDEX.TS

```

import { Grid, Aggregate } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Aggregate);
let grid: Grid = new Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd' },
    { field: 'Freight', headerText: 'Freight', width: 150, format: 'C2'
},
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
  ],
  height: 230,
  aggregates: [{
    columns: [{
      type: 'Sum',
      field: 'Freight',
      format: 'C2',
      footerTemplate: 'Sum: ${Sum}'
    }]
  },
  {
    columns: [{
      type: 'Max',

```

```

        field: 'Freight',
        format: 'C2',
        footerTemplate: 'Max: ${Max}'
    }
}
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <div id="toolbar-template">
      <div id="refresh" title="Refresh">

```

```

        <button class="e-btn e-flat">
            <span class="e-btn-icon e-icons e-refresh"></span>
        </button>
    </div>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to place aggregates on top of the Grid

By default, the aggregated values are placed at the bottom of the footer section. It is possible to place the aggregated values at the top of the header. This is achieved by using the [dataBound](#) event, [getHeaderContent](#), and [getFooterContent](#) methods of the Grid.

In the following sample, the footer element is appended to the header element using the [getHeaderContent](#) and [getFooterContent](#) methods in the [dataBound](#) event.

INDEX.TS

```

import { Grid, Aggregate } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Aggregate);
let grid: Grid = new Grid({
    dataSource: data,
    dataBound: dataBound,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'Freight', headerText: 'Freight', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 210,
    aggregates: [{
        columns: [{
            type: 'Sum',
            field: 'Freight',
            footerTemplate: 'Sum: ${Sum}'
        }]
    },
    {
        columns: [{
            type: 'Max',
            field: 'Freight',
            footerTemplate: 'Max: ${Max}'
        }]
    }
    ]
});
grid.appendTo('#Grid');
function dataBound() {

```

```

grid.getHeaderContent().append(grid.getFooterContent());
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <div id="toolbar-template">
      <div id="refresh" title="Refresh">
        <button class="e-btn e-flat">
          <span class="e-btn-icon e-icons e-refresh"></span>
        </button>
      </div>
    </div>
  </div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Group and caption aggregate in EJ2 JavaScript Grid control

Group and caption aggregate values are calculated from the current group items. If [groupFooterTemplate](#) is provided, the aggregate values are displayed in the group footer cells; and if [groupCaptionTemplate](#) is provided, aggregate values are displayed in the group caption cells.

INDEX.TS

```

import { Grid, Group, Aggregate } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group, Aggregate);
let grid: Grid = new Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { showDropArea: false, columns: ['ShipCountry'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'OrderDate', headerText: 'Order Date', width: 120, format:
'yMd' },
        { field: 'Freight', headerText: 'Freight', width: 150, format: 'C2'
},
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 290,
    aggregates: [{
        columns: [{
            type: 'Sum',
            field: 'Freight',
            format: 'C2',
            groupFooterTemplate: 'Sum: ${Sum}'
        }]
    },
    {
        columns: [{
            type: 'Max',
            field: 'Freight',
            format: 'C2',
            groupCaptionTemplate: 'Max: ${Max}'
        }]
    }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <div id="toolbar-template">
      <div id="refresh" title="Refresh">
        <button class="e-btn e-flat">
          <span class="e-btn-icon e-icons e-refresh"></span>
        </button>
      </div>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The aggregate values must be accessed inside the template using their corresponding [type](#) name.

Custom aggregate in EJ2 JavaScript Grid control

To calculate the aggregate value with your own aggregate functions, use the custom aggregate option. To use custom aggregation, specify the [type](#) as `Custom`, and provide the custom aggregate function in the [customAggregate](#) property.

The custom aggregate function will be invoked with different arguments for total and group aggregations.

- **Total aggregation:** The custom aggregate function will be called with the whole data and current [AggregateColumn](#)

object.

- **Group aggregation:** This will be called with the current group details and [AggregateColumn](#) object.

INDEX.TS

```

import { Grid, Aggregate } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Aggregate);
let customAggregateFn = (data: Object, aggColumn: Object) =>
data.result.filter((item: Object) => item[aggColumn.columnName] ===
'Brazil').length;
let grid: Grid = new Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'Freight', headerText: 'Freight', width: 150, format: 'C2'
},
    { field: 'ShipCountry', headerText: 'Ship Name', width: 150 }
  ],
  height: 268,
  aggregates: [{
    columns: [{
      type: 'Custom',
      customAggregate: customAggregateFn,
      columnName: 'ShipCountry',
      footerTemplate: 'Brazil Count: ${Custom}'
    }]
  }]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <div id="toolbar-template">
      <div id="refresh" title="Refresh">
        <button class="e-btn e-flat">
          <span class="e-btn-icon e-icons e-refresh"></span>
        </button>
      </div>
    </div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To access the custom aggregate value inside the template, use the key as **Custom**.

Show the count of distinct values in aggregate row

You can calculate the aggregate value with your own aggregate functions. To use custom aggregation, specify the [type](#) as **Custom**, and provide the custom aggregate function in the [customAggregate](#) property.

In this below demo, we have show the count of distinct value for **ShipCountry** column by using custom aggregate.

INDEX.TS

```

import { Grid, Page, Aggregate } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { DataManager, Query, DataUtil } from '@syncfusion/ej2-data';
Grid.Inject(Page, Aggregate);
let customAggregateFn = function() {
  let results = new DataManager(this.currentViewData).executeLocal(new
  Query().select(['ShipCountry']));
  let distinct = DataUtil.distinct(results, 'ShipCountry', true);
  return distinct.length;
}
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
    { field: "ShipCountry", headerText: "Ship Country", width: 150 }
  ],
  height: 220,
  aggregates: [{
    columns: [{
      type: 'Custom',
      customAggregate: customAggregateFn,
      columnName: 'ShipCountry',
      footerTemplate: 'Distinct Count: ${Custom}'
    }]
  }]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Reactive aggregate in EJ2 JavaScript Grid control

Auto update aggregate value in batch editing

When using batch editing, the aggregate values will be refreshed on every cell save. The footer, group footer, and group caption aggregate values will be refreshed.

Adding a new record to the grouped grid will not refresh the aggregate values.

INDEX.TS

```
import { Grid, Aggregate, Edit, Toolbar, Group, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Aggregate, Edit, Toolbar, Group, Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  pageSettings: { pageSize: 6 },
  toolbar: ['Delete', 'Update', 'Cancel'],
  editSettings: { allowEditing: true, allowDeleting: true, mode: 'Batch' },
},
{
  allowGrouping: true,
  groupSettings: { showDropArea: false, columns: ['ShipCountry'] },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', isPrimaryKey: true,
      textAlign: 'Right', width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'Freight', headerText: 'Freight', width: 150, format: 'C2' },
  ],
  { field: 'ShipCountry', headerText: 'Ship Name', width: 150 }
],
  height: 268,
  aggregates: [{
    columns: [{
      type: 'Sum',
      field: 'Freight',
      format: 'C2',
      footerTemplate: 'Sum : ${Sum}'
    }]
  },
  {
    columns: [{
      type: 'Sum',
      field: 'Freight',
      format: 'C2',
      groupCaptionTemplate: 'Average : ${Average}'
    }]
  },
  {
    columns: [{
      type: 'Sum',
      field: 'Freight',
      format: 'C2',
      groupFooterTemplate: 'Sum : ${Sum}'
    }]
  }
]
});
```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
    <div id="toolbar-template">
      <div id="refresh" title="Refresh">
        <button class="e-btn e-flat">
          <span class="e-btn-icon e-icons e-refresh"></span>
        </button>
      </div>
    </div>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Refresh aggregate values in inline editing

By default, reactive aggregate update is not supported by inline and dialog edit modes as it is not feasible to anticipate the value change event for every editor. But, you can refresh the aggregates manually in the inline edit mode using the refresh method of aggregate module.

In the following code, the input event for the Freight column editor has been registered and the aggregate value has been refreshed manually.

INDEX.TS

```
import { Grid, Aggregate, Edit, Toolbar, Group, Page } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Aggregate, Edit, Toolbar, Group, Page);
let selectedRecord : Object = {};
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging:true,
    pageSettings:{pageSize:7},
    toolbar: ['Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowDeleting: true, mode: 'Normal'
},
    actionBegin:(args:any)=>{
        if(args.requestType === 'beginEdit'){
            selectedRecord ={};
            selectedRecord = args.rowData;
        };
    },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', isPrimaryKey:true,
        textAlign: 'Right', width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'Freight', headerText: 'Freight', editType: 'numericedit',
        format: 'C2', edit: { params: { change: (args :any) => {
            let gridObj =
            document.getElementById('Grid')['ej2_instances'][0];
            selectedRecord['Freight'] = args.value; // Set the edited value
            to aggregate column
            gridObj.aggregateModule.refresh(selectedRecord) // Refresh
            aggregates using edited data
        }
        }
        }, width: 150},
        { field: 'ShipCountry', headerText: 'Ship Name', width: 150 }
    ],
    height: 268,
    aggregates: [{
```



```

        columns:[{
            type:'Sum',
            field:'Freight',
            format:'C2',
            footerTemplate:'Sum : ${Sum}'
        }]
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>

```

```

        <div id="toolbar-template">
            <div id="refresh" title="Refresh">
                <button class="e-btn e-flat">
                    <span class="e-btn-icon e-icons e-refresh"></span>
                </button>
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print in EJ2 JavaScript Grid control

To print the Grid, use the [print](#) method from grid instance. The print option can be displayed on the [toolbar](#) by adding the `print` toolbar item.

INDEX.TS

```

import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Print'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 272
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Page setup

Some of the print options cannot be configured through JavaScript code. So, you have to customize the layout, paper size, and margin options using the browser page setup dialog. Please refer to the following links to know more about the browser page setup:

- [Chrome](#)
- [Firefox](#)
- [Safari](#)
- [IE](#)

Print using an external button

To print the grid from an external button, invoke the [print](#) method.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 270
});
grid.appendTo('#Grid');
let printBtn: Button = new Button();
printBtn.appendTo('#print');
document.getElementById('print').addEventListener('click', () => {
  grid.print();
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="print">Print</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print the visible page

By default, the grid prints all the pages. To print the current page alone, set the [printMode](#) to **CurrentPage**.

INDEX.TS

```

import { Grid, Toolbar, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, Page);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Print'],
    printMode: 'CurrentPage',
    allowPaging: true,
    pageSettings: { pageSize: 6 },
    columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
        }
    </style>

```

```

        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print the hierarchy grid

By default, the grid will be print the master and expanded child grids alone. you can change the print option by using the [hierarchyPrintMode](#) property. The available options are,

Mode	Behavior
Expanded	Prints the master grid with expanded child grids.
All	Prints the master grid with all the child grids.
None	Prints the master grid alone.

INDEX.TS

```

import { Grid, DetailRow, Toolbar } from '@syncfusion/ej2-grids';
import { data, employeeData } from './datasource.ts';
Grid.Inject(DetailRow, Toolbar);
let grid: Grid = new Grid({
    dataSource: employeeData,
    toolbar: ['Print'],
    hierarchyPrintMode: 'All',
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        dataSource: data,

```

```

        queryString: 'EmployeeID',
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ]
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

<div id="container">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print the master detail grid

The Grid has the option to visualize details of a record in another Grid in a master and detailed manner. By default, Grid will print the master grid alone. Instead of this, it is possible to print both the master and detail grids by using the [beforePrint](#) event of the Grid.

In the following sample, the detail grid is added to the `element` argument of the `beforePrint` event, resulting in both the master and detail grids being printed on the page.

INDEX.TS

```

import { Grid, RowSelectEventArgs, Selection, Toolbar } from
 '@syncfusion/ej2-grids';
import { data, customerData } from './datasource.ts';
Grid.Inject(Selection, Toolbar);
type carType = { CustomerID: string; CustomerName: string; ContactName:
 string; };
let names: string[] = ['AROUT', 'BERGS', 'BLONP', 'CHOPS', 'ERNSH'];
let masterdata: Object = customerData.filter((e: carType) =>
 names.indexOf(e.CustomerID) !== -1);
let mastergrid: Grid = new Grid({
  dataSource: masterdata,
  toolbar: ['Print'],
  selectedRowIndex: 1,
  beforePrint: beforePrint,
  columns: [
    { field: 'ContactName', headerText: 'Customer Name', width: 150 },
    { field: 'CompanyName', headerText: 'Company Name', width: 150 },
    { field: 'Address', headerText: 'Address', width: 150 },
    { field: 'Country', headerText: 'Country', width: 130 },
  ],
  rowSelected: rowSelected,
});
mastergrid.appendTo('#MasterGrid');
function rowSelected(args: RowSelectEventArgs): void {
  let selectedRecord: carType = args.data as carType;
  grid.dataSource = data.filter((record: carType) => record.CustomerName
  === selectedRecord.ContactName).slice(0, 5);
  document.getElementById('key').innerHTML = selectedRecord.ContactName;
}
function beforePrint(args) {
  let customEle = document.createElement('div');
  customEle.innerHTML = document.getElementsByClassName('e-
  statustext')[0].innerHTML + grid.element.innerHTML;
  customEle.appendChild(document.createElement('br'));
}

```

```

        args.element.append(customEle);
    }
    let grid: Grid = new Grid({
        allowSelection: false,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', width: 100, textAlign:
'Right' },
            { field: 'Freight', headerText: 'Freight', width: 100, format: 'C2',
type: 'number' },
            { field: 'ShipName', headerText: 'Ship Name', width: 200 },
            { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
            { field: 'ShipAddress', headerText: 'Ship Address', width: 200 },
        ],
    });
    grid.appendTo('#DetailGrid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

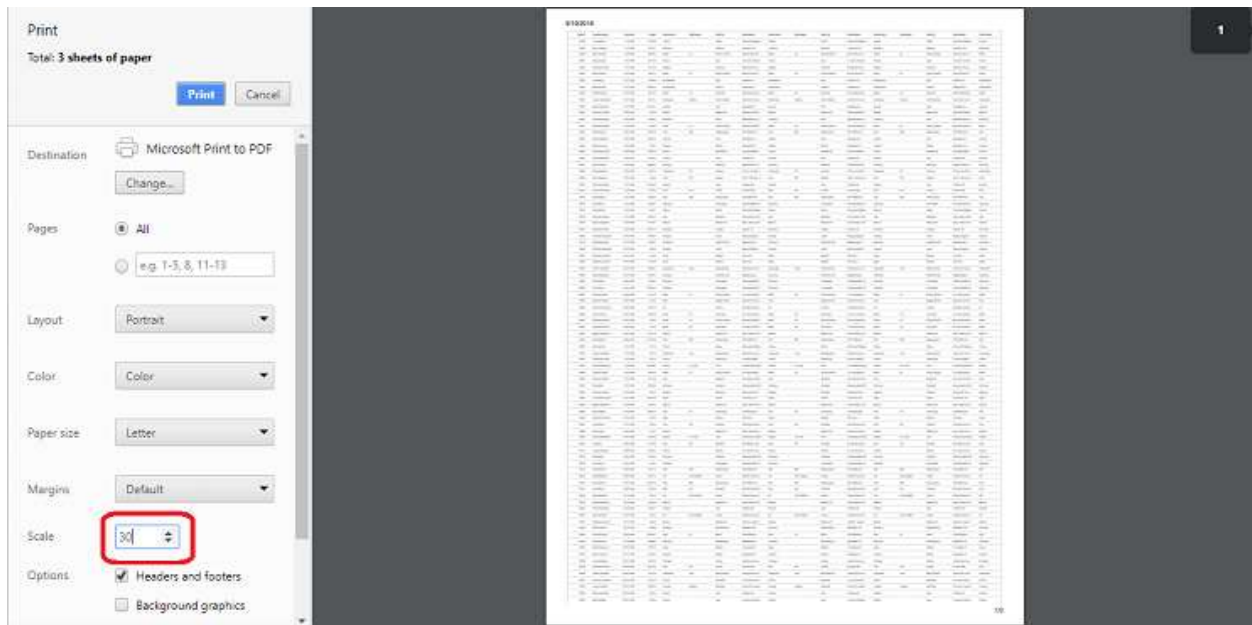
    <div id="container">
        <p class="e-mastertext">Master Grid</p>
        <div id="MasterGrid">
        </div>
        <p></p><div class="e-statustext"> Showing orders of Customer: <b
id="key"></b></div><p></p>
        <div id="DetailGrid">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print large number of columns

By default, the browser uses A4 as page size option to print pages and to adapt the size of the page the browser print preview will auto-hide the overflowed contents. Hence grid with large number of columns will cut off to adapt the print page.

To show large number of columns when printing, adjust the scale option from print option panel based on your content size.



Show or Hide columns while Printing

You can show a hidden column or hide a visible column while printing the grid using [toolbarClick](#) and [printComplete](#) events.

In the `toolbarClick` event, based on `args.item.id` as `grid_print`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `printComplete` event, We have reversed the state back to the previous state.

In the below example, we have `CustomerID` as a hidden column in the grid. While printing, we have changed `CustomerID` to visible column and `ShipCity` as hidden column.

INDEX.TS

```
import { Grid, DetailRow, Toolbar, Page } from '@syncfusion/ej2-grids';
import { data, employeeData } from './datasource.ts';
Grid.Inject(DetailRow, Toolbar, Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  toolbar: ['Print'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', visible:
false, width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  pageSettings: { pageSizes: true, pageSize: 6 },
  toolbarClick: function(args) {
    for (var i = 0; i < this.columns.length; i++) {
      if (this.columns[i].field == "CustomerID") {
        this.columns[i].visible = true;
      }
      else if (this.columns[i].field == "ShipCity") {
        this.columns[i].visible = false;
      }
    }
  },
  printComplete: function(args) {
    for (var i = 0; i < this.columns.length; i++) {
      if (this.columns[i].field == "CustomerID") {
        this.columns[i].visible = false;
      }
      else if (this.columns[i].field == "ShipCity") {
        this.columns[i].visible = true;
      }
    }
  }
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
```

```

    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations of Printing Large Data

When grid contains large number of data, printing all the data at once is not a best option for the browser performance. Because to render all the DOM elements in one page will produce performance issues in the browser. It leads to browser slow down or browser hang. Grid have option to handle large number of data by Virtualization. However while printing, it is not possible to use virtualization for rows and columns.

If printing of all the data is still needed, we suggest to Export the grid to **Excel** or **CSV** or **Pdf** and then print it from another non-web based application.

See Also

- [How to Print the expanded state grid from all pages](#)
- [How to print only selected records in grid](#)

Adaptive in EJ2 JavaScript Grid control

The Grid user interface (UI) was redesigned to provide an optimal viewing experience and improve usability on small screens. This interface will render the filter, sort, column chooser, column menu(supports only when the **rowRenderingMode** as Horizontal) and edit dialogs adaptively and have an option to render the grid row elements in the vertical direction.

Render adaptive dialogs

When we enable the [enableAdaptiveUI](#) property, the grid will render the filter, sort, and edit dialogs in full screen for a better user experience. This behavior is demonstrated in the below demo.

INDEX.TS

```
import { Grid, Filter, Sort, Edit, Toolbar, Page } from '@syncfusion/ej2-grids';
import { Browser } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
Grid.Inject(Filter, Sort, Edit, Toolbar, Page);
let grid: Grid = new Grid({
  dataSource: data,
  enableAdaptiveUI: true,
  allowPaging: true,
  allowSorting: true,
  allowFiltering: true,
  filterSettings: { type: 'Excel' },
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel', 'Search'],
  editSettings: { allowAdding: true, allowEditing: true, allowDeleting:
true, mode: 'Dialog' },
  height: '100%',
  load: () => {
    grid.adaptiveDlgTarget = document.getElementsByClassName('e-mobile-content')[0] as HTMLElement;
  },
  columns: [
    { field: 'SNO', headerText: 'S NO', isPrimaryKey: true, width: 150,
validationRules: { required: true, number: true } },
    { field: 'Model', headerText: 'Model Name', width: 200, editType:
"dropdownedit", validationRules: { required: true } },
    { field: 'Developer', headerText: 'Developer', filter: { type :
'Menu' }, width: 200, validationRules: { required: true } },
    { field: 'ReleaseDate', headerText: 'Released Date', type: 'date',
editType: "datepickeredit", format: 'yMMM', width: 200 },
    { field: 'AndroidVersion', headerText: 'Android Version', filter: {
type : 'CheckBox' }, width: 200, validationRules: { required: true } }
  ]
});
grid.appendTo('#adaptivebrowser');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="e-adaptive-demo e-bigger">
    <div class="e-mobile-layout">
      <div class="e-mobile-content">
        <div id="adaptivebrowser"></div>
      </div>
    </div>
    <br>
    <div class="datalink">Source:
      <a
href="https://en.wikipedia.org/wiki/List_of_Android_smartphones"
target="_blank">Wikipedia: List of Android smartphones</a>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Vertical row rendering

The grid will render the row elements in vertical order while setting the [rowRenderingMode](#) property value as **Vertical**.

INDEX.TS

```

import { Grid, Filter, Sort, Edit, Toolbar, Aggregate, Page } from
 '@syncfusion/ej2-grids';
import { Browser } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
Grid.Inject(Filter, Sort, Edit, Toolbar, Aggregate, Page);
let grid: Grid = new Grid({
  dataSource: data,
  enableAdaptiveUI: true,
  rowRenderingMode: 'Vertical',
  allowPaging: true,
  allowSorting: true,
  allowFiltering: true,
  filterSettings: { type: 'Excel' },
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel', 'Search'],
  editSettings: { allowAdding: true, allowEditing: true, allowDeleting:
true, mode: 'Dialog' },
  height: '100%',
  load: () => {
    grid.adaptiveDlgTarget = document.getElementsByClassName('e-mobile-
content')[0] as HTMLElement;
  },
  columns: [
    { field: 'SNO', headerText: 'S NO', isPrimaryKey: true, width: 150,
validationRules: { required: true, number: true } },
    { field: 'Model', headerText: 'Model Name', width: 200, editType:
"dropdownedit", validationRules: { required: true } },
    { field: 'Developer', headerText: 'Developer', filter: { type :
'Menu' }, width: 200, validationRules: { required: true } },
    { field: 'ReleaseDate', headerText: 'Released Date', type: 'date',
editType: "datepickeredit", format: 'yMMM', width: 200 },
    { field: 'AndroidVersion', headerText: 'Android Version', filter: {
type : 'CheckBox' }, width: 200, validationRules: { required: true } }
  ],
  aggregates: [{
    columns: [{
      type: 'Count',
      field: 'Model',
      footerTemplate: 'Total Models: ${Count}'
    }]
  }]
});
grid.appendTo('#verticalrender');

```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="e-adaptive-demo e-bigger">
    <div class="e-mobile-layout">
      <div class="e-mobile-content">
        <div id="verticalrender"></div>
      </div>
    </div>
    <br>
    <div class="datalink">Source:
      <a
href="https://en.wikipedia.org/wiki/List_of_Android_smartphones"
target="_blank">Wikipedia: List of Android smartphones</a>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

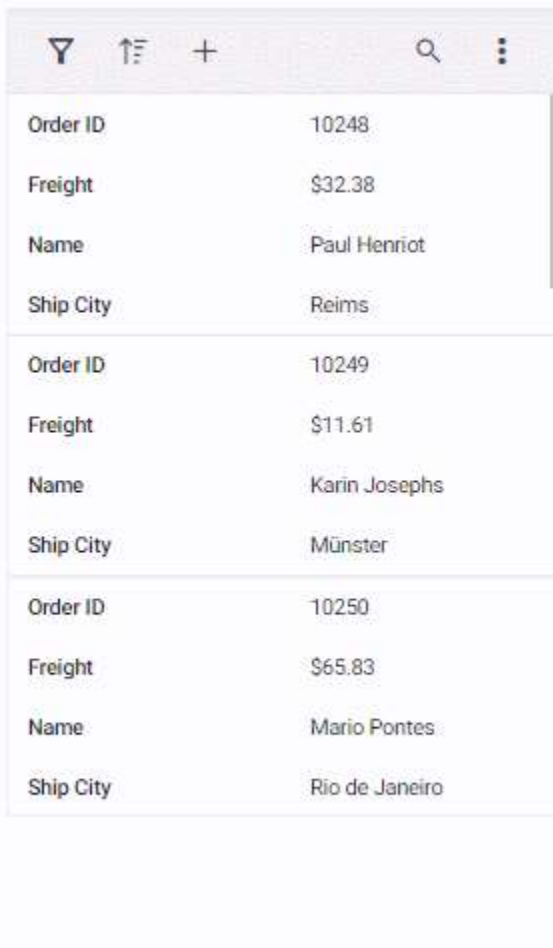
```
</body></html>
```

* [enableAdaptiveUI](#) property must be enabled for vertical row rendering.

Supported features by vertical row rendering

The following features are only supported in vertical row rendering:

- Paging, including Page size dropdown
- Sorting
- Filtering
- Selection
- Dialog Editing
- Aggregate
- Infinite scroll
- Toolbar - Options like **Add**, **Filter**, **Sort**, **Edit**, **Delete**, **Search**, and **Toolbar template** are available when their respective features are enabled. The toolbar dynamically includes a three-dotted icon, containing additional features like **ColumnChooser**, **Print**, **PdfExport**, **ExcelExport**, or **CsvExport**, once these features are enabled. Please refer to the following snapshot.



Order ID	10248
Freight	\$32.38
Name	Paul Henriot
Ship City	Reims
Order ID	10249
Freight	\$11.61
Name	Karin Josephs
Ship City	Münster
Order ID	10250
Freight	\$65.83
Name	Mario Pontes
Ship City	Rio de Janeiro

A snapshot of the adaptive grid displaying enabled paging along with a pager dropdown.

<div> <div></div> <div></div> <div></div> </div>	
Order ID	10248
Freight	\$32.38
Name	Vins et alcools Chevalier
Ship City	Reims
Order ID	10249
Freight	\$11.61
Name	Toms Spezialitäten
Ship City	Münster
Order ID	10250
Freight	\$65.83
<div> <div> <div></div> <div></div> <div></div> </div> <div>1 / 6</div> <div> <div></div> <div></div> </div> <div>Items per page: 5</div> </div>	

The Column Menu feature, which includes grouping, sorting, autofit, filter, and column chooser, is exclusively supported for the Grid in **Horizontal** [rowRenderingMode](#).

Hierarchy grid in EJ2 JavaScript Grid control

The Grid allows display of table data in a hierarchical structure to visualize relations between parent and child records. This feature is enabled by defining the [childGrid](#) and [childGrid.queryString](#). The [childGrid](#) describes the options of grid and the [childGrid.queryString](#) describes the relation between parent and child grids.

To use hierarchical binding, inject the [DetailRow](#) module in the grid.

INDEX.TS

```
import { Grid, DetailRow } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { employeeData } from './employeeData.ts';
Grid.Inject(DetailRow);
let grid: Grid = new Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
```

```

        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        dataSource: data,
        queryString: 'EmployeeID',
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
    },
    height: 315
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
    </style>

```

```

        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* Grid supports n level of child grids.

* Hierarchical binding is not supported when [DetailTemplate](#) is enabled.

ExpandAll by external button

By default, grid renders in collapsed state. You can expand all child grid rows by invoking the [expandAll](#) method, and collapse all grid rows by invoking the [collapseAll](#) through an external button.

INDEX.TS

```

import { Grid, DetailRow } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { data, employeeData } from './datasource.ts';
Grid.Inject(DetailRow);
let expandBtn: Button = new Button();
expandBtn.appendTo('#expandall');
let collapseBtn: Button = new Button();
collapseBtn.appendTo('#collapseall');
let grid: Grid = new Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },

```

```

        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        dataSource: data,
        queryString: 'EmployeeID',
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
    },
    height: 265
});
grid.appendTo('#Grid');
document.getElementById('expandall').addEventListener('click', () => {
    grid.detailRowModule.expandAll();
});
document.getElementById('collapseall').addEventListener('click', () => {
    grid.detailRowModule.collapseAll();
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="expandall">Expand All</button>
        <button id="collapseall">Collapse All</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Expand child grid initially

You can expand a particular child grid at initial rendering by invoking the [expand](#) method in the [dataBound](#) event.

INDEX.TS

```

import { Grid, DetailRow } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { employeeData } from './employeeData.ts';
Grid.Inject(DetailRow);
let grid: Grid = new Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        dataSource: data,
        queryString: 'EmployeeID',
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ]
    }
});

```

```

    ],
    },
    height: 315,
    dataBound: onDataBound
});
grid.appendTo('#Grid');
function onDataBound(): void {
    this.detailRowModule.expand(1); // initial expand 1 child Grid.
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {

```



```

        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamically load child grid data

You can dynamically load child grid dataSource by using the [load](#) event. This [load](#) event triggers when the child grid is expanded for the first time.

INDEX.TS

```

import { Grid, DetailRow } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { employeeData } from './employeeData.ts';
Grid.Inject(DetailRow);
let grid: Grid = new Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        queryString: 'EmployeeID',
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
        load: onLoad
    },

```

```

        height: 315
    });
    grid.appendTo('#Grid');
    function onLoad(args: Object): void {
        this.dataSource = data; // assign data source for child grid.
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;

```

```

        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Bind hierarchy grid with different field

By default, Parent and child grid relation will be maintained by `queryString` property. We should use the same field name to map both parent and child grid. To achieve parent and child relation with different fields, we need to change the mapping value in the child grid `load` event.

In the below sample, we have bound the child and parent grid with different fields. Parent grid field name as `EmployeeID` and the child grid field name as `ID`. We need to define the mapping value of `parentKeyFieldValue` from the parent row data in the child grid `load` event.

INDEX.TS

```

import { Grid, DetailRow } from '@syncfusion/ej2-grids';
import { data } from './childdata.ts';
import { employeeData } from './employeeData.ts';
Grid.Inject(DetailRow);
let grid: Grid = new Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        dataSource: data,
        queryString: 'ID',
        columns: [
            { field: 'ID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 },

```

```

        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    load: onLoad,
},
height: 315
});
grid.appendTo('#Grid');
function onLoad(): void {
    this.parentDetails.parentKeyFieldValue =
    this.parentDetails.parentRowData['EmployeeID'];
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
    </style>

```

```

    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding Record in ChildGrid

Parent and child grid are related by [queryString](#) field value. To maintain this relation in newly added record, You need to set value for [queryString](#) field in the added data by the [actionBegin](#) event.

In the below demo, **EmployeeID** field relates the parent and child grids. To add a new record in child grid, We have to set the **EmployeeID** field with parent record's [queryString](#) field value in the [actionBegin](#) event.

INDEX.TS

```

import { Grid, DetailRow, AddEventArgs, Edit, Toolbar } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { employeeData } from './employeeData.ts';
Grid.Inject(DetailRow, Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        dataSource: data,
        queryString: 'EmployeeID',
        toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    }
});

```

```

        editSettings: { allowEditing: true, allowAdding: true,
allowDeleting: true },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', isPrimaryKey:true,
textAlign: 'Right', width: 120 },
            { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', allowEditing:false, width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
        actionBegin: actionBegin
    },
    height: 315
});
grid.appendTo('#Grid');
function actionBegin(args: AddEventArgs): void {
    if (args.requestType === "add") {
        // `parentKeyFieldValue` refers to the queryString field value of
        the parent record.
        args.data.EmployeeID = this.parentDetails.parentKeyFieldValue;
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamically bind data to child grid based on parent row Data

By default, the [childGrid.queryString](#) describes the relationship between parent and child grids and visualizes the data in a hierarchical structure. Instead of the `queryString` property, we can dynamically bind the datasource to the `childGrid` based on the parent row data using the [detailDataBound](#) event of the grid.

While expanding the child Grid, the `detailDataBound` event will be triggered. In this event, based on the EmployeeID column value of parent row data, filter the equally matched data from the `childGrid` datasource using the `DataManager` plugin and bind the filtered data as a datasource to the `childGrid`. This can be demonstrated by the following sample.

INDEX.TS

```

import { Grid, DetailRow, DetailDataBoundEventArgs } from '@syncfusion/ej2-
grids';
import { data } from './datasource.ts';

```

```

import { employeeData } from './employeeData.ts';
import { DataManager, Query } from '@syncfusion/ej2-data';
Grid.Inject(DetailRow);
let orderData = data;
let grid: Grid = new Grid({
    dataSource: employeeData,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        columns: [
            { field: 'OrderID', headerText: 'Order ID', isPrimaryKey: true,
textAlign: 'Right', width: 120 },
            { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right' width: 120 },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
    },
    height: 315,
    detailDataBound: (args: DetailDataBoundEventArgs) => {
        let empIdValue =
args.childGrid.parentDetails.parentRowData.EmployeeID;
        let matchedData = new DataManager(orderData).executeLocal(
            new Query().where('EmployeeID', 'equal', empIdValue, true)
        );
        args.childGrid.query = new Query();
        args.childGrid.dataSource = matchedData;
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

State Persistence

State persistence in EJ2 JavaScript Grid control

State persistence refers to the Grid's state maintained in the browser's [localStorage](#) even if the browser is refreshed or if you move to the next page within the browser. State persistence stores grid's model object in the local storage when the [enablePersistence](#) is defined as true.

Restore initial Grid state

When the [enablePersistence](#) property is set to **true**, the Grid will keep its state even if the page is reloaded. In some cases, you may be required to retain the Grid in its initial state. The Grid will not retain its initial state now since the [enablePersistence](#) property has been enabled.

You can achieve this by destroying the grid after disabling the `enablePersistence` property and clearing the local storage data, as shown in the sample below.

INDEX.TS

```
import { Grid, Filter, ActionEventArgs, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Filter, Page);
let grid: Grid = new Grid({
    dataSource: data,
    enablePersistence: true,
    allowFiltering: true,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 230
});
grid.appendTo('#Grid');
document.getElementById('restore').onclick = () => {
    grid.enablePersistence = false;
    window.localStorage.setItem("gridGrid", "");
    grid.destroy();
    //reloads the page
    location.reload();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="restore">Restore to initial state</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Maintaining custom query in a persistent state

The grid does not maintain the query params after page load event when the [enablePersistence](#) is set to true. This is because the grid refreshes its query params for every page load. You can maintain the custom query params by resetting the [addParams](#) method in the [actionBegin](#) event.

INDEX.TS

```

import { Grid, Filter, ActionEventArgs, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Filter, Page);
let grid: Grid = new Grid({
    dataSource: data,

```

```

allowFiltering: true,
allowPaging: true,
enablePersistence: true,
columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
],
actionBegin: actionHandler,
height: 230
});
grid.appendTo('#Grid');
function actionHandler(args: ActionEventArgs) {
    this.query.addParams('$filter', 'EmployeeID eq 1');
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add a new column in persisted columns list

The Grid columns can be persisted when the [enablePersistence](#) property is set to true. If you want to add the new columns with the existing persist state, you can use the Grid inbuilt method such as `column.push` and call the [refreshColumns\(\)](#) method for UI changes. Please refer to the following sample for more information.

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({
    dataSource: data,
    enablePersistence: true,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 230
});
grid.appendTo('#Grid');
document.getElementById('add').onclick = () => {
    let obj = { field: "Freight", headerText: 'Freight', width: 120 }
    grid.columns.push(obj as any); //you can add the columns by using the
Grid columns method
    grid.refreshColumns();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="add">Add columns</button>
        <div id="Grid"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Get or set local storage value in EJ2 JavaScript Grid control

If the [enablePersistence](#) property is set to true, the grid property value is saved in the `window.localStorage` for reference. You can get/set the `localStorage` value by using the `getItem/setItem` method in the `window.localStorage`.

```
`ts
//get the Grid model.

let value: string = window.localStorage.getItem('gridGrid'); //"gridGrid" is component name +
component id.

let model: Object = JSON.parse(model);
`ts

//set the Grid model.

window.localStorage.setItem('gridGrid', JSON.stringify(model)); //"gridGrid" is component name +
component id.
```

Prevent to persist in EJ2 JavaScript Grid control

Prevent columns from persisting

When the [enablePersistence](#) property is set to true, the Grid properties such as [Grouping](#), [Paging](#), [Filtering](#), [Sorting](#), and [Columns](#) will persist. You can use the `addOnPersist` method to prevent these Grid properties from persisting.

The following example demonstrates how to prevent Grid columns from persisting. In the [dataBound](#) event of the Grid, you can override the `addOnPersist` method and remove the columns from the key list given for persistence.

Note: When the `enablePersistence` property is set to true, the Grid features such as column template, column formatter, header text, and value accessor will not persist.

INDEX.TS

```
import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({
    dataSource: data,
    enablePersistence: true,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 230,
    dataBound: dataBound
});
```

```

grid.appendTo('#Grid');
function dataBound(args: any) {
    let cloned = this.addOnPersist;
    this.addOnPersist = function (key: any) {
        key = key.filter((item: string) => item !== "columns");
        return cloned.call(this, key);
    };
}
document.getElementById('add').onclick = () => {
    let obj = { field: "Freight", headerText: 'Freight', width: 120 }
    grid.columns.push(obj as any); //you can add the columns by using the
Grid columns method
    grid.refreshColumns();
};
document.getElementById('remove').onclick = () => {
    grid.columns.pop();
    grid.refreshColumns();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="add">Add columns</button>
        <button id="remove">Remove columns</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add to persist in EJ2 JavaScript Grid control

Persist the column template, header template and header Text

By default, the Grid properties such as column template, header text, header template, column formatter, and value accessor will not persist when [enablePersistence](#) is set to true. Because the column template and header text can be customized at the application level.

If you wish to restore all these column properties, then you can achieve it by cloning the grid's columns property using JavaScript Object's assign method and storing this along with the persist data manually. While restoring the settings, this column object must be assigned to the grid's columns property to restore the column settings as demonstrated in the following sample.

INDEX.TS

```

import { Grid, Page, Filter } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Filter, Page);
let grid: Grid = new Grid({
    dataSource: data,
    enablePersistence: true,
    allowFiltering: true,
    allowPaging: true,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150,
headerTemplate: '#customertemplate' },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150, template:
'#template' },
    ],
    height: 230
});
grid.appendTo('#Grid');

```

```

let savedProperties: any;
document.getElementById('restore').onclick = () => {
    savedProperties = JSON.parse(grid.getPersistData());
    var gridColumnState = Object.assign([], grid.getColumns());
    savedProperties.columns.forEach((col: {
        field: any;
        headerText: any;
        template: any;
        headerTemplate: any;
    }) => {
        let headerText = gridColumnState.find((colColumnsState) =>
colColumnsState.field === col.field)['headerText'];
        let colTemplate = gridColumnState.find((colColumnsState) =>
colColumnsState.field === col.field)['template'];
        let headerTemplate = gridColumnState.find((colColumnsState) =>
colColumnsState.field === col.field)['headerTemplate'];
        col.headerText = 'Text Changed';
        col.template = colTemplate;
        col.headerTemplate = headerTemplate; //likewise you can restore
required column properties as per your wants.
    })
    );
    console.log(savedProperties);
    grid.setProperties(savedProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-column:before {
        content: '\e815';
    }
    .e-header:before {
        content: '\ea9a';
    }
</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
    <script id="template" type="text/x-template">
        <a rel='nofollow'
href=https://en.wikipedia.org/wiki/${ShipName}><span class="e-icons e-column"></span></a>
    </script>

    <script id="customertemplate" type="text/x-template">
        <span class="e-icons e-header" ></span>
        Customer ID
    </script>

    <div id="container">
        <button id="restore">Restore</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tool Bar

Tool bar in EJ2 JavaScript Grid control

The Grid provides ToolBar support to handle grid actions. The [toolbar](#) property accepts either the collection of built-in toolbar items and [ItemModel](#) objects for custom toolbar items or HTML element ID for toolbar template.

To use ToolBar, inject [Toolbar](#) module in the grid.

Enable or disable toolbar items

You can enable/disable toolbar items by using the **enableItems** method.

INDEX.TS

```

import { Grid, Toolbar, Group } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from './datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
Grid.Inject(Toolbar, Group);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Expand', 'Collapse'],
    allowGrouping: true,
    groupSettings: { columns: ['CustomerID'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 200,
    toolbarClick: clickHandler
});
grid.appendTo('#Grid');
let enable: Button = new Button({}, '#enable');
let disable: Button = new Button({}, '#disable');
enable.element.onclick = () => {
    grid.toolbarModule.enableItems(['Grid_Collapse', 'Grid_Expand'],
true); // enable toolbar items.
};
disable.element.onclick = () => {
    grid.toolbarModule.enableItems(['Grid_Collapse', 'Grid_Expand'],
false); // disable toolbar items.
};
function clickHandler(args: ClickEventArgs): void {
    if (args.item.id === 'Grid_Collapse') { // grid_Collapse is component id
+ '_' + toolbar item name.
        grid.groupModule.collapseAll();
    }
    if (args.item.id === 'Grid_Expand') {
        grid.groupModule.expandAll();
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="enable" class="e-flat">Enable</button>
        <button id="disable" class="e-flat">Disable</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add toolbar at the bottom of Grid

You can add the Grid toolbar component at the bottom of Grid using the ['created'](#) event.

INDEX.TS

```

import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';

```

```

Grid.Inject(Toolbar);
let grid: Grid = new Grid({
    dataSource: data.slice(0,8),
    toolbar: ['Print', 'Search'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 200,
    created: () => {
        let toolbar: HTMLElement = grid.element.querySelector('.e-toolbar');
        grid.element.appendChild(toolbar);
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Toolbar Component](#)
- [How to set overflow mode in Grid toolbar](#)

Tool bar items in EJ2 JavaScript Grid control

Built-in toolbar items

Built-in toolbar items execute standard actions of the grid, and it can be added by defining the [toolbar](#) as a collection of built-in items. It renders the button with icon and text.

The following table shows built-in toolbar items and its actions.

Built-in Toolbar Items	Actions
-----	-----
Add	Adds a new record.
Edit	Edits the selected record.
Update	Updates the edited record.
Delete	Deletes the selected record.
Cancel	Cancels the edit state.
Search	Searches the records by the given key.
Print	Prints the grid.
ExcelExport	Exports the grid to Excel.
PdfExport	Exports the grid to PDF.
WordExport	Exports the grid to Word.

| ColumnChooser | Choose the column's visibility.|

INDEX.TS

```
import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Print', 'Search'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 270
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```



```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The [toolbar](#) has options to define both built-in and custom toolbar items.

Show only icons in built-in toolbar items

By default, the built-in toolbar items render as buttons with an icon and text. It is possible to hide the text and show only the icon using the following CSS style.

```

.e-toolbar .e-tbar-btn-text, .e-toolbar .e-toolbar-items .e-toolbar-item .e-tbar-btn-text {
display: none;
}

```

This is demonstrated in the following sample:

INDEX.TS

```
import { Grid, Toolbar, Edit } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, Edit);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number', isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 140, type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 270
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<style>
    .e-toolbar .e-tbar-btn-text,
    .e-toolbar .e-toolbar-items .e-toolbar-item .e-tbar-btn-text {
        display: none;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom toolbar items

Custom toolbar items can be added by defining the [toolbar](#) as a collection of [ItemModels](#). Actions for this customized toolbar items are defined in the [toolbarClick](#) event.

By default, Custom toolbar items are in position **Left**. You can change the position by using the [align](#) property. In the below sample, we have applied position **Right** for the **Collapse All** toolbar item.

INDEX.TS

```

import { Grid, Toolbar, Group } from '@syncfusion/ej2-grids';
import { EmitType } from '@syncfusion/ej2-base';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, Group);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'expandall') {
        grid.groupModule.expandAll();
    }
    if (args.item.id === "collapseall") {
        grid.groupModule.collapseAll();
    }
};
let grid: Grid = new Grid({
    dataSource: data,
    allowGrouping: true,
    groupSettings: { columns: ['CustomerID'] },
    toolbar: [{ text: 'Expand All', tooltipText: 'Expand All', prefixIcon:
'e-expand', id: 'expandall' }, { text: 'Collapse All', tooltipText:

```

```

'collection All', prefixIcon: 'e-collapse', id: 'collapseall', align: 'Right'
}],
    toolbarClick: clickHandler,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 200,
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

    <style>
        .e-icons.e-collapse::before {
            content: "\e834";

```

```

    }
    .e-icons.e-expand::before {
        content: "\e83d";
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* The [toolbar](#) has options to define both built-in and custom toolbar items.

* If a toolbar item does not match the built-in items, it will be treated as a custom toolbar item.

Custom toolbar items are elements that are added to the toolbar, such as input, button, span, etc. In order to focus these items using the TAB or Shift + Tab keys, the `e-toolbar-item-focus` class should be used in the elements. If the `e-toolbar-item-focus` class is not used in the custom toolbar item elements, the focus of the item will be skipped when using the TAB or Shift + Tab keys.

Both built-in and custom items in toolbar

Grid have an option to use both built-in and custom toolbar items at same time.

In the below example, `Add`, `Edit`, `Delete`, `Update`, `Cancel` are built-in toolbar items and `Click` is custom toolbar item.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { EmitType } from '@syncfusion/ej2-base';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'Click') {
        alert("Custom toolbar click...");
    }
};
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel', { text: 'Click',
        tooltipText: 'Click', prefixIcon: 'e-expand', id: 'Click' }],

```

```

        toolbarClick: clickHandler,
        editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
            { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
            { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
        ],
        height: 265
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
    </style>

```

```

    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom toolbar component in a specific position

By default, Custom toolbar items are in the Left position. You can change the position by using the [align](#) property. In the following sample, we have applied the Right position for the Collapse All toolbar item, Left for the Expand All toolbar item, and Center for the Search toolbar item.

INDEX.TS

```

import { Grid, Toolbar, Group } from '@syncfusion/ej2-grids';
import { EmitType } from '@syncfusion/ej2-base';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { employeeData } from './datasource.ts';
Grid.Inject(Toolbar, Group);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    if (args.item.id === 'expandall') {
        grid.groupModule.expandAll();
    }
    if (args.item.id === "collapseall") {
        grid.groupModule.collapseAll();
    }
};
let grid: Grid = new Grid({
    dataSource: employeeData,
    allowGrouping: true,
    groupSettings: { columns: ['FirstName'] },

```

```

        toolbar: [{ text: 'Expand All', tooltipText: 'Expand All', prefixIcon:
'e-expand', id: 'expandall', align: 'Left'}, { text: 'Collapse All',
tooltipText: 'collection All', prefixIcon: 'e-collapse', id: 'collapseall',
align: 'Right' }, { text: 'Search', align: 'Center'}]],
        toolbarClick: clickHandler,
        columns: [
            { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120, type: 'number' },
            { field: 'FirstName', width: 140, headerText: 'First Name', type:
'string' },
            { field: 'Country', headerText: 'Country', textAlign: 'Right',
width: 120, country: 'string' },
            { field: 'PostalCode', headerText: 'Postal Code', textAlign:
'Right', width: 140, type: 'string' }
        ],
        height: 200,
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

    <style>

```



```

        .e-icons.e-collapse::before {
            content: "\e834";
        }
        .e-icons.e-expand::before {
            content: "\e83d";
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom tool bar in EJ2 JavaScript Grid control

Custom toolbar is used to customize the whole toolbar. It can be added by defining **toolbarTemplate** as an HTML element ID.

Actions for this toolbar template items are defined in the [toolbarClick](#) event.

INDEX.TS

```

import { Grid, Toolbar, Group } from '@syncfusion/ej2-grids';
import { EmitType } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
Grid.Inject(Toolbar, Group);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
    var target: Element =
    (<HTMLElement>args.originalEvent.target).closest('.e-toolbar-item');
    if (target.id === 'collapse') {
        grid.groupModule.collapseAll();
    }
};
let grid: Grid = new Grid({
    dataSource: data,
    toolbarTemplate: '#toolbar-template',
    toolbarClick: clickHandler,
    allowGrouping: true,
    groupSettings: { columns: ['CustomerID'] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },

```

```

        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 200
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

<div id="Grid"></div>
<div id="toolbar-template">
  <div>
    <div id="collapse" title="Collapse">
      <button class="e-btn e-flat">
        <span class="e-btn-icon e-icons e-collapse"></span>
      </button>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render drop-down list in custom toolbar

You can create your own ToolBar items in the Grid. It can be added by defining the [toolbar](#) as HTML element ID. Actions for this ToolBar template items are defined in the [toolbarClick](#).

To include components in the ToolBar, please ensure the following steps:

Step 1:

Initialize the template for your custom component. Using the following code add the DropDownList component to the ToolBar.

```

<div id='toolbar-template'>
  <div id='dropdown' style="margin-top:5px">
    <input type="text" tabindex="1" id='ddlelement' />
  </div>
</div>

```

Step 2:

To render the DropDownList component, use the [dataBound](#) event of the Grid.

- You can select the grid row index based on the selected data in the DropDownList. The output will appear as follows.

INDEX.TS

```

import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
let rowIndex: any = [0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14];

```

```

Grid.Inject(Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbarTemplate: '#toolbar-template',
    dataBound: dataBound,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 200
});
grid.appendTo('#Grid');
function dataBound(): void {
    let dropDownListObject: DropDownList = new DropDownList({
        // set the data to dataSource property
        dataSource: rowIndex,
        change: change,
        popupHeight :200
    });
    dropDownListObject.appendTo('#ddlelement');
}
function change(args: ChangeEventArgs): void {
    grid.selectRow(<number>args.itemData);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar-template">
            <div id="dropdown" style="margin-top:5px">
                <input type="text" tabindex="1" id="ddlelement">
            </div>
        </div>
        <div id="Grid"></div>
    </div>
<style>
    .orientationcss .e-headercelldiv {
        transform: rotate(90deg);
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pdf Export

Pdf export in EJ2 JavaScript Grid control

PDF export allows exporting Grid data to PDF document. You need to use the [pdfExport](#) method for exporting. To enable PDF export in the grid, set the [allowPdfExport](#) as true.

To use PDF export, inject the PdfExport module in grid.

INDEX.TS

```

import { Grid, Page, Toolbar, PdfExport } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,

```

```

        allowPdfExport: true,
        toolbar: ['PdfExport'],
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
            { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
            { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
        ],
        height: 230
    });
    grid.appendTo('#Grid');
    grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_pdfexport') {
        grid.pdfExport();
    }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show spinner while exporting

You can show/ hide spinner component while exporting the grid using `showSpinner/ hideSpinner` methods. You can use `toolbarClick` event to show spinner before exporting and hide a spinner in the `pdfExportComplete` or `excelExportComplete` event after the exporting.

In the `toolbarClick` event, based on the parameter `args.item.id` as `Gridpdfexport` or `Gridexcelexport` we can call the `showSpinner` method from grid instance.

In the `pdfExportComplete` or `excelExportComplete` event, We can call the `hideSpinner` method.

In the below demo, we have rendered the default spinner component when exporting the grid.

INDEX.TS

```

import { Grid, Page, Toolbar, PdfExport, ExcelExport } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport, ExcelExport);
let grid: Grid = new Grid({

```

```

        dataSource: data,
        allowPaging: true,
        allowPdfExport: true,
        allowExcelExport: true,
        toolbar: ['PdfExport', 'ExcelExport'],
        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
            { field: 'CustomerID', width: 140, headerText: 'Customer ID',
visible: false },
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
            { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
width: 140 }
        ],
        height: 230
    });
    grid.appendTo('#Grid');
    grid.toolbarClick = (args: Object) => {
        if (args['item'].id === 'Grid_pdfexport') {
            grid.showSpinner();
            grid.pdfExport();
        }
        if (args['item'].id === 'Grid_excelexport') {
            grid.showSpinner();
            grid.excelExport();
        }
    }
    grid.pdfExportComplete = () => {
        grid.hideSpinner();
    }
    grid.excelExportComplete = () => {
        grid.hideSpinner();
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom data source

PDF export provides an option to define datasource dynamically before exporting. To export data dynamically, define the `dataSource` in `exportProperties`

INDEX.TS

```

import { Grid, Toolbar, Page, PdfExport, PdfExportProperties } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    let exportProperties: PdfExportProperties = {
      dataSource: data,
    };
    grid.pdfExport(exportProperties);
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Passing additional parameters to the server when exporting

You can pass the additional parameter in the `query` property by invoking `addParams` method. In the `toolbarClick` event, you can define params as key and value pair so it will receive at the server side when exporting.

In the below example, we have passed `recordcount` as `12` using `addParams` method.

INDEX.TS

```

import { Grid, Page, Toolbar, PdfExport, ExcelExport } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { Query } from '@syncfusion/ej2-data';
Grid.Inject(Page, Toolbar, PdfExport, ExcelExport);
let queryClone: Object;
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  allowExcelExport: true,
  toolbar: ['PdfExport', 'ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID',
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
width: 140 }
  ],
  height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    queryClone = grid.query;
    grid.query = new Query().addParams("recordcount", "12");
    grid.pdfExport();
  }
  if (args['item'].id === 'Grid_excelexport') {
    queryClone = grid.query;
    grid.query = new Query().addParams("recordcount", "12");
    grid.excelExport();
  }
}
grid.pdfExportComplete = () => {
  grid.query = queryClone;
}
grid.excelExportComplete = () => {
  grid.query = queryClone;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Passing the selected records to the server using ajax request via custom toolbar button click

You can pass the selected records to the server with the help of an ajax request. In the `toolbarClick` event, you can get the selected records using the [getSelectedRecords](#) method and pass the selected records to the server using the **data** property of the ajax.

`ts

```

import { Grid, Page, Toolbar } from '@syncfusion/ej2-grids';
import { Ajax } from '@syncfusion/ej2-base';

Grid.Inject(Page, Toolbar);

let grid: Grid = new Grid({
  dataSource: data,
  allowSelection: true,
  toolbar: ['Selected'],
  columns: [
    { type: 'checkbox', width: 50, },
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 120},
    { field: 'CustomerID', width: 140, headerText: 'CustomerID'},
    { field: 'ShipCity', headerText: 'Ship Country', width: 140 }
  ],
  toolbarClick: function (args) {
    if (args.item.text === 'Selected') {
      var selectedRecord = this.getSelectedRecords();
      var ajax = new Ajax({
        url: "/Home/SelectRecord",
        type: "POST",
        contentType: "application/json",
        data: JSON.stringify(selectedRecord),
        successHandler: function (response) {
          console.log(JSON.parse(response));
        },
        failure: function (response) {
          alert(response);
        }
      });
    }
  }
});

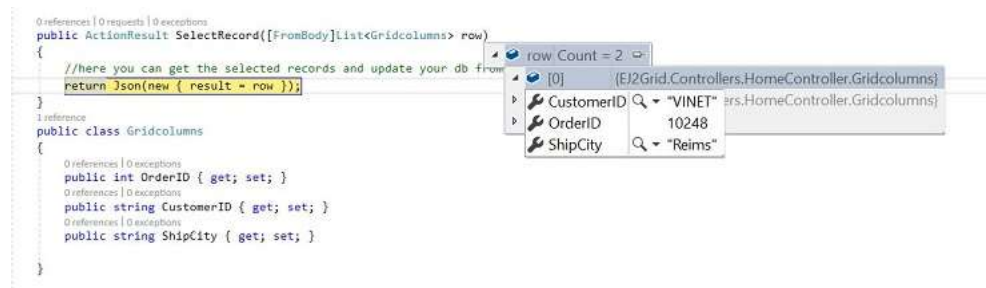
```

```

}
});
ajax.send();
}
},
});
grid.appendTo('#Grid');
`

```

The selected record details are bound to the `row` parameter. Please refer to the following screenshot.



[See Also](#)

- [How to Exporting Grid in Cordova application](#)

Export multiple grids in EJ2 JavaScript Grid control

The PDF export provides an option to export multiple grids to the same or different pages of a PDF file. Each grid is identified by its unique ID. You can specify which grid to export by listing their **IDs** in the `exportGrids` property.

Same page

PDF exporting provides support for exporting multiple grids on the same page. To export the grids on the same page, define `multipleExport.type` as **AppendToPage** in `exportProperties`. It also has an option to provide blank space between the grids. This blank space can be defined by using `multipleExport.blankSpace` property.

INDEX.TS

```

import { Grid, Page, Toolbar, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-grids';
import { data, employeeData } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let firstGrid: Grid = new Grid({
    dataSource: data.slice(0, 5),
    allowPaging: true,
    allowPdfExport: true,
    exportGrids: ['FirstGrid', 'SecondGrid'],
    toolbar: ['PdfExport'],
    columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
    ],
});
firstGrid.appendTo('#FirstGrid');
let secondGrid: Grid = new Grid({
    dataSource: employeeData.slice(0, 5),
    allowPaging: true,
    allowPdfExport: true,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'FirstName', width: 140, headerText: 'First Name', type:
'string' },
        { field: 'LastName', width: 140, headerText: 'Last Name', type:
'string' },
        { field: 'City', headerText: 'City', width: 120 },
    ],
});
secondGrid.appendTo('#SecondGrid');
firstGrid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'FirstGrid_pdfexport') {
        let appendPdfExportProperties: PdfExportProperties = {
            multipleExport: { type: "AppendToPage", blankSpace: 10 }
        };
        firstGrid.pdfExport(appendPdfExportProperties, true);
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <p><b>First Grid</b></p>
        <div id="FirstGrid"></div>
        <p><b>Second Grid</b></p>
        <div id="SecondGrid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

New page

PDF export functionality enables the exporting of multiple grids into separate pages (each grid on a new page) within the PDF file. To achieve this, you can specify `multipleExport.type` as `NewPage` in `exportProperties`.

INDEX.TS

```

import { Grid, Page, Toolbar, PdfExport, PdfExportProperties } from
 '@syncfusion/ej2-grids';
import { data, employeeData } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let firstGrid: Grid = new Grid({
    dataSource: data.slice(0, 5),
    allowPaging: true,
    allowPdfExport: true,
    exportGrids: ['FirstGrid', 'SecondGrid'],
    toolbar: ['PdfExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },

```

```

        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
    ],
});
firstGrid.appendTo('#FirstGrid');
let secondGrid: Grid = new Grid({
    dataSource: employeeData.slice(0, 5),
    allowPaging: true,
    allowPdfExport: true,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120, type: 'number' },
        { field: 'FirstName', width: 140, headerText: 'First Name', type:
'string' },
        { field: 'LastName', width: 140, headerText: 'Last Name', type:
'string' },
        { field: 'City', headerText: 'City', width: 120 },
    ],
});
secondGrid.appendTo('#SecondGrid');
firstGrid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'FirstGrid_pdfexport') {
        let appendPdfExportProperties: PdfExportProperties = {
            multipleExport: { type: 'NewPage' }
        };
        firstGrid.pdfExport(appendPdfExportProperties, true);
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <p><b>First Grid</b></p>
        <div id="FirstGrid"></div>
        <p><b>Second Grid</b></p>
        <div id="SecondGrid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pdf export options in EJ2 JavaScript Grid control

[Export current page](#)

PDF export provides an option to export the current page into PDF. To export current page, define the `exportType` to `CurrentPage`.

INDEX.TS

```

import { Grid, Toolbar, Page, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    ]
});

```

```

        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_pdfexport') {
        let exportProperties: PdfExportProperties = {
            exportType: 'CurrentPage'
        };
        grid.pdfExport(exportProperties);
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
    </style>

```

```

        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export the selected records only

You can export the selected records data by passing it to `exportProperties.dataSource` Property in the `toolbarClick` event.

In the below exporting demo, We can get the selected records using `getSelectedRecords` method and pass the selected data to `PdfExport` or `excelExport` property.

INDEX.TS

```

import { Grid, Page, Toolbar, PdfExport, Filter, ExcelExport } from
 '@syncfusion/ej2-grids';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { data } from './datasource.ts';
import { DataManager } from '@syncfusion/ej2-data';
Grid.Inject(Page, Toolbar, PdfExport, Filter, ExcelExport);
let grid: Grid = new Grid(
    {
        dataSource: data,
        allowPdfExport: true,
        allowExcelExport: true,
        allowPaging: true,
        allowFiltering: true,
        selectionSettings: {type: 'Multiple'},
    }
);

```

```

        toolbar: ['PdfExport', 'ExcelExport'],
        pageSettings: { pageCount: 5, pageSize: 5 },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', width: 120,
textAlign: 'Right' },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'OrderDate', headerText: 'Order Date', width: 130,
format: 'yMd', textAlign: 'Right' },
            { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right'
},
            { field: 'ShipCountry', visible: false, headerText: 'Ship
Country', width: 150 }
        ],
    });
grid.appendTo('#Grid');
grid.toolbarClick = (args: ClickEventArgs) => {
    if (args.item.id === 'Grid_pdfexport') {
        let pdfdata = grid.getSelectedRecords();
        let exportProperties = {
            dataSource: pdfdata,
        };
        grid.pdfExport(exportProperties);
    }
    else if (args.item.id === 'Grid_excelexport') {
        let exceldata = grid.getSelectedRecords();
        let exportProperties = {
            dataSource: exceldata,
        };
        grid.pdfExport(exportProperties);
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export filtered data only

You can export the filtered data by defining the resulted data in `exportProperties.dataSource` before export.

In the below Pdf exporting demo, We have gotten the filtered data by applying filter query to the grid data and then defines the resulted data in `exportProperties.dataSource` and pass it to `pdfExport` method.

INDEX.TS

```
import { Grid, Page, Toolbar, PdfExport, Filter } from '@syncfusion/ej2-
grids';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { data } from './datasource.ts';
import { DataManager } from '@syncfusion/ej2-data';
Grid.Inject(Page, Toolbar, PdfExport, Filter);
let grid: Grid = new Grid(
    {
        dataSource: data,
        allowPdfExport: true,
        allowPaging: true,
        allowFiltering: true,
        toolbar: ['PdfExport'],
        pageSettings: { pageCount: 5, pageSize: 5 },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', width: 120,
textAlign: 'Right' },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'OrderDate', headerText: 'Order Date', width: 130,
format: 'yMd', textAlign: 'Right' },
            { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right'
},
            { field: 'ShipCountry', visible: false, headerText: 'Ship
Country', width: 150 }
        ],
    });
grid.appendTo('#Grid');
grid.toolbarClick = (args: ClickEventArgs) => {
    if (args.item.id === 'Grid_pdfexport') {
        let pdfdata;
        let query = grid.renderModule.data.generateQuery(); // get grid
corresponding query
        for (var i = 0; i < query.queries.length; i++) {
            if (query.queries[i].fn == 'onPage') {
                query.queries.splice(i, 1); // remove page query to get
all records
                break;
            }
        }
        let fdata = new DataManager({ json: data
}).executeQuery(query).then((e: any) => {
            pdfdata = <Object[]>e.result; // get all filtered records
            let exportProperties = {
                dataSource: pdfdata,
            };
            grid.pdfExport(exportProperties);
        }).catch((e) => true);
    }
}
```


INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Export hidden columns

PDF export provides an option to export hidden columns of Grid by defining the `includeHiddenColumn` as `true`.

INDEX.TS

```
import { Grid, Toolbar, Page, PdfExport, PdfExportProperties } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'ShipCountry', width: 140, headerText: 'Ship Country',
visible: false },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    let exportProperties: PdfExportProperties = {
      includeHiddenColumn: true
    };
    grid.pdfExport(exportProperties);
  }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide columns

You can show a hidden column or hide a visible column while exporting the grid using [toolbarClick](#) and [pdfExportComplete](#) events.

In the `toolbarClick` event, based on `args.item.id` as `Grid_pdfexport`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `pdfExportComplete` event, We have reversed the state back to the previous state.

In the below example, we have `CustomerID` as a hidden column in the grid. While exporting, we have changed `CustomerID` to visible column and `ShipCity` as hidden column.

INDEX.TS

```
import { Grid, Page, Toolbar, PdfExport } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID',
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
width: 140 }
  ],
  height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    grid.columns[1].visible = false;
    grid.columns[3].visible = true;
    grid.pdfExport();
  }
}
grid.pdfExportComplete = () => {
  grid.columns[1].visible = false;
  grid.columns[3].visible = true;
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change page orientation

Page orientation can be changed Landscape(Default Portrait) for the exported document using the `exportProperties`.

INDEX.TS

```

import { Grid, Page, Toolbar, PdfExport, PdfExportProperties } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
' string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    let exportProperties: PdfExportProperties = {
      pageOrientation: 'Landscape',
    };
    grid.pdfExport(exportProperties);
  }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Change page size

Page size can be customized for the exported document using the `exportProperties`.

Supported page sizes are:

- Letter
- Note
- Legal
- A0
- A1
- A2
- A3
- A5
- A6
- A7
- A8
- A9
- B0
- B1
- B2
- B3
- B4
- B5
- Archa
- Archb
- Archc
- Archd
- Arche
- Flsa
- HalfLetter
- Letter11x17
- Ledger

INDEX.TS

```
import { Grid, Page, Toolbar, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
```



```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_pdfexport') {
        let exportProperties: PdfExportProperties = {
            pageSize: 'Letter'
        };
        grid.pdfExport(exportProperties);
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Define file name

You can assign the file name for the exported document by defining `fileName` property in [PdfExportProperties](#).

INDEX.TS

```

import { Grid, Page, Toolbar, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowPdfExport: true,
    pageSettings: { pageSize: 6 },
    toolbar: ['PdfExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },

```

```

        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
        'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
        width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
        width: 140, format: 'yMd' }
    ],
    height: 220
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_pdfexport') {
        let exportProperties: PdfExportProperties = {
            fileName: "new.pdf"
        };
        grid.pdfExport(exportProperties);
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {

```

```

        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Font customization

Default fonts

By default, grid uses **Helvetica** font in the exported document. You can change the default font by using **pdfExportProperties.theme** property. The available default fonts are,

- Helvetica
- TimesRoman
- Courier
- Symbol
- ZapfDingbats

The code example for changing default font,

```
`ts
```

```
import { PdfStandardFont, PdfFontFamily, PdfFontStyle } from '@syncfusion/ej2-pdf-export';
```

```
...
```

```
let pdfExportProperties: PdfExportProperties = {
  theme: {
    header: {font: new PdfStandardFont(PdfFontFamily.TimesRoman, 11, PdfFontStyle.Bold),
    caption: { font: new PdfStandardFont(PdfFontFamily.TimesRoman, 9) },
    record: { font: new PdfStandardFont(PdfFontFamily.TimesRoman, 10) }
  }
};
`
```

Add custom font

You can change the default font of Grid header, content and caption cells in the exported document by using `pdfExportProperties.theme` property.

In the following example, we have used Advent Pro font to export the grid with Hungarian fonts.

INDEX.TS

```
import { Grid, Page, Toolbar, PdfExport, Group, PdfExportProperties } from
 '@syncfusion/ej2-grids';
import { data, adventProFont } from './datasource.ts';
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';
Grid.Inject(Page, Toolbar, PdfExport, Group);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  allowGrouping: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Rendelés azonosító', textAlign:
 'Right', width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Ügyfél-azonosító',
 type: 'string' },
    { field: 'Freight', headerText: 'fuvar', textAlign: 'Right', width:
 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Rendelés dátuma', textAlign:
 'Right', width: 140, format: 'yMd' }
  ],
  height: 172
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    let pdfExportProperties: PdfExportProperties = {
      theme: {
        header: {font: new PdfTrueTypeFont(adventProFont, 12) },
        caption: { font: new PdfTrueTypeFont(adventProFont, 10) },
        record: { font: new PdfTrueTypeFont(adventProFont, 9) }
      }
    };
    grid.pdfExport(pdfExportProperties);
  }
}
```

```
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

PdfTrueTypeFont accepts base 64 format of the Custom Font.

Export grid as blob

The Grid offers an option to export the data as a **Blob** instead of downloading it as a file in the browser. To export the grid as a Blob, set the **isBlob** parameter to **true** in the [pdfExport](#) method. The grid returns the promise of a blob in the [pdfExportComplete](#) event.

The following example demonstrates how to obtain the blob data of the exported grid by executing the promise in the **pdfExportComplete** event.

INDEX.TS

```
import { Grid, Page, Toolbar, PdfExport, PdfExportCompleteArgs } from
 '@syncfusion/ej2-grids';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  toolbarClick: toolbarClick,
  pdfExportComplete: pdfExportComplete,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 230
});
grid.appendTo('#Grid');
function toolbarClick(args: ClickEventArgs) {
  if (args.item.id === 'Grid_pdfexport') {
    // pass fourth parameter as true to get the blob data of exported
grid
    grid.pdfExport(null, null, null, true);
  }
}
function pdfExportComplete(args: PdfExportCompleteArgs) {
  // execute the promise to get the blob data
  args.promise.then((e: { blobData: Blob }) => {
    console.log(e.blobData);
  });
};
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pdf cell style customization in EJ2 JavaScript Grid control

Conditional cell formatting

Grid cells in the exported PDF can be customized or formatted using [pdfQueryCellInfo](#) event. In this event, we can format the grid cells of exported PDF document based on the column cell value.

In the below sample, we have set the background color for **Freight** column in the exported document by **args.cell** and **backgroundColor** property.

INDEX.TS

```
import { Grid, Page, Toolbar, PdfExport } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    grid.pdfExport();
  }
}
grid.pdfQueryCellInfo = (args: Object) => {
  if (args.column.field === 'Freight') {
    if (args.value < 30) {
      args.style = { backgroundColor: '#99ffcc' };
    }
    else if (args.value < 60) {
      args.style = { backgroundColor: '#ffffb3' };
    }
    else {
      args.style = { backgroundColor: '#ff704d' };
    }
  }
}
grid.queryCellInfo = (args: Object) => {
  if (args.column.field === 'Freight') {
    if (args.data['Freight'] < 30) {
      args.cell.bgColor = '#99ffcc';
    }
    else if (args.data['Freight'] < 60) {
      args.cell.bgColor = '#ffffb3';
    }
    else {
      args.cell.bgColor = '#ff704d';
    }
  }
}
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Theme

PDF export provides an option to include theme for exported PDF document.

To apply theme in exported PDF, define the `theme` in `exportProperties`.

INDEX.TS

```

import { Grid, Toolbar, Page, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_pdfexport') {
        let exportProperties: PdfExportProperties = {
            theme: {
                header: {
                    fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true, border: { color: '#64FA50', lineStyle: 'Thin' }
                },
                record: {

```

```

        fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true
    },
    caption: {
        fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true
    }
    }
};
grid.pdfExport (exportProperties);
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {

```

```

        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, material theme is applied to exported PDF document.

Adding header and footer in EJ2 JavaScript Grid control

You can customize text, page number, line, page size and changing orientation in header and footer.

Write a text in header and footer

You can add text either in Header or Footer of exported PDF document.

```
`ts
```

```

let exportProperties: PdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{
type: 'Text',
value: "Northwind Traders",
position: { x: 0, y: 50 },
style: { textBrushColor: '#000000', fontSize: 13 }
}
]
}
}

```

```
},
]
}
`ts
```

Draw a line in header and footer

you can add line either in Header or Footer of the exported PDF document.

Supported line styles:

- dash
- dot
- dashdot
- dashdotdot
- solid

```
`ts
let exportProperties: PdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{
type: 'Line',
style: { penColor: '#000080', penSize: 2, dashStyle: 'Solid' },
points: { x1: 0, y1: 4, x2: 685, y2: 4 }
}
]
}
}
`ts
```

Add page number in header and footer

you can add page number either in Header or Footer of exported PDF document.

Supported page number types:

- LowerLatin - a, b, c,
- UpperLatin - A, B, C,
- LowerRoman - i, ii, iii,
- UpperRoman - I, II, III,
- Number - 1,2,3.

```
`ts
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'PageNumber',
        pageNumberType: 'Arabic',
        format: 'Page {$current} of {$total}', //optional
        position: { x: 0, y: 25 },
        style: { textBrushColor: '#ffff80', fontSize: 15, hAlign: 'Center' }
      }
    ]
  }
}
```

Insert an image in header and footer

Image (Base64 string) can be added in the exported document in header/footer using the `exportProperties`.

```
`ts
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'Image',
        src: image,
        position: { x: 40, y: 10 },
        size: { height: 100, width: 250 },
      }
    ]
  }
}
```

```

}
,

```

The below code illustrates the pdf export customization.

INDEX.TS

```

import { Grid, Page, Toolbar, PdfExport, PdfExportProperties } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { image } from './image.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_pdfexport') {
        let pdfExportProperties: PdfExportProperties = {
            header: {
                fromTop: 0,
                height: 130,
                contents: [
                    {
                        type: 'Image',
                        src: image,
                        position: { x: 40, y: 10 },
                        size: { height: 100, width: 250 },
                    }
                ]
            },
            footer: {
                fromBottom: 160,
                height: 150,
                contents: [
                    {
                        type: 'PageNumber',
                        pageNumberType: 'Arabic',
                        format: 'Page { $current } of { $total }',
                        position: { x: 0, y: 25 },
                        style: { textBrushColor: '#ffff80', fontSize: 15 }
                    }
                ]
            }
        }
    }
}

```



```

    }
    };
    grid.pdfExport(pdfExportProperties);
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>

```

```

    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
      <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Repeat column header on every page

By default, column header will be placed on the first page of the pdf document but you can display column header on every page using `repeatHeader` property of `pdfGrid`.

In the below sample, we have enabled `repeatHeader` property in `pdfHeaderQueryCellInfo` event to show the header on every page.

INDEX.TS

```

import { Grid, Toolbar, Page, PdfExport, PdfExportProperties } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120},
    { field: 'CustomerID', width: 140, headerText: 'Customer ID'},
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'ShipName', headerText: 'ShipName', textAlign: 'Right',
width: 140}
  ],
  height: 230,
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    grid.pdfExport();
  }
}

```

```

}
grid.pdfHeaderQueryCellInfo = (args: Object) => {
    args.cell.row.pdfGrid.repeatHeader = true;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>

```

```

</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting hierarchy grid in EJ2 JavaScript Grid control

The grid have an option to export the hierarchy grid to pdf document. By default, grid will exports the master grid with expanded child grids alone. you can change the exporting option by using the PdfExportProperties.hierarchyExportMode property. The available options are,

Mode	Behavior
-----	-----
Expanded	Exports the master grid with expanded child grids.
All	Exports the master grid with all the child grids.
None	Exports the master grid alone.

INDEX.TS

```

import { Grid, DetailRow, Toolbar, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-grids';
import { data, employeeData } from './datasource.ts';
Grid.Inject(DetailRow, Toolbar, PdfExport);
let grid: Grid = new Grid({
    dataSource: employeeData,
    toolbar: ['PdfExport'],
    allowPdfExport: true,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
        { field: 'FirstName', headerText: 'First Name', width: 150 },
        { field: 'City', headerText: 'City', width: 150 },
        { field: 'Country', headerText: 'Country', width: 150 }
    ],
    childGrid: {
        dataSource: data,
        queryString: 'EmployeeID',
        columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
    },
    toolbarClick: toolbarClick
});
grid.appendTo('#Grid');
function toolbarClick(args) {
    if (args['item'].id === 'Grid_pdfexport') {
        let exportProperties: PdfExportProperties = {
            hierarchyExportMode: "Expanded"
        };
        grid.pdfExport(exportProperties);
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting grid with templates in EJ2 JavaScript Grid control

The grid offers the option to export the column, detail, and caption templates to a PDF document. The template contains images, hyperlinks, and customized text.

Exporting with column template

The PDF export functionality allows you to export Grid columns that include images, hyperlinks, and custom text to an PDF document.

In the following sample, the hyperlinks and images are exported to PDF using [hyperlink](#) and [image](#) properties in the [pdfQueryCellInfo](#) event.

PDF Export supports base64 string to export the images.

INDEX.TS

```

import { Grid, Toolbar, PdfExport, PdfQueryCellInfoEventArgs } from
'@syncfusion/ej2-grids';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { employeeData } from './datasource.ts';
Grid.Inject(Toolbar, PdfExport);
let grid: Grid = new Grid({
    dataSource: employeeData,
    allowPdfExport: true,
    toolbar: ['PdfExport'],
    columns: [
        {
            headerText: 'Employee Image', textAlign: 'Center',
            template: '#imageTemplate', width: 150
        },
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 125 },
        { field: 'FirstName', headerText: 'Name', width: 120 },
        { field: 'EmailID', headerText: 'Email ID', template:
'#mailTemplate', width: 170 }
    ],
    toolbarClick: toolbarClick,

```

```

        pdfQueryCellInfo: pdfQueryCellInfo,
        height: 273
    });
    grid.appendTo('#Grid');
    function toolbarClick(args: ClickEventArgs) {
        if (args.item.id === 'Grid_pdfexport') {
            grid.pdfExport();
        }
    }
    function pdfQueryCellInfo(args: PdfQueryCellInfoEventArgs): any {
        if (args.column.headerText === 'Employee Image') {
            args.image = ({
                base64: args.data['EmployeeImage'],
                height: 50,
                width: 50,
            } as any);
        }
        if (args.column.headerText === 'Email ID') {
            args.hyperLink = {
                target: 'mailto:' + args.data['EmailID'],
                displayText: args.data['EmailID'],
            };
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">





  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="imageTemplate" type="text/x-template">
            <div class="image">
                
            </div>
        </script>
        <script id="mailTemplate" type="text/x-template">
            <div class="link">
                <a href="mailto:{EmailID}">{EmailID}</a></div>
            </script>
            <div id="Grid"></div>
        </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```


PDF Export		
Employee Image	Name	Email ID
	Nancy	nancy@domain.com
	Andrew	andrew@domain.com
	Janet	janet@domain.com
	Margaret	margaret@domain.com

Exporting with detail template

By default, the grid will export the parent grid with expanded detail rows alone. Change the exporting option by using the `PdfExportProperties.hierarchyExportMode` property. The available options are:

| Mode | Behavior |

|-----|-----|

| Expanded | Exports the parent grid with expanded detail rows. |

| All | Exports the parent grid with all the detail rows. |

| None | Exports the parent grid alone. |

The detail rows in the exported PDF can be customized or formatted using the [exportDetailTemplate](#) event. In this event, the detail rows of the PDF document are formatted in accordance with their parent row details.

In the following sample, the detail row content is formatted by specifying the [columnCount](#), [columnHeader](#), and [rows](#) properties using its [parentRow](#) details. This allows for the creation of detail rows in the PDF document. Additionally, custom styles can be applied to specific cells using the [style](#) property.

If `columnCount` is not provided, the columns in the detail row of the PDF grid will be generated based on the count of the `columnHeader/rows` first row's [cells](#).

When using [rowSpan](#), it is essential to provide the cell's [index](#) for proper functionality.

INDEX.TS

```
import { Grid, DetailRow, Toolbar, PdfExport, PdfExportProperties,
ExportDetailTemplateEventArgs } from '@syncfusion/ej2-grids';
import { employeeData } from './datasource.ts';
Grid.Inject(DetailRow, Toolbar, PdfExport);
let grid: Grid = new Grid({
  dataSource: employeeData,
  detailTemplate: '#detailtemplate',
  toolbar: ['PdfExport'],
  allowPdfExport: true,
  toolbarClick: toolbarClick,
  exportDetailTemplate: exportDetailTemplate,
  columns: [
    { field: 'Category', headerText: 'Category', width: 140 },
    { field: 'ProductID', headerText: 'Product ID', width: 120 },
    { field: 'Status', headerText: 'Status', width: 120 }
  ],
  height: 273
});
grid.appendTo('#Grid');
function toolbarClick(args) {
  if (args['item'].id === 'Grid_pdfexport') {
    let exportProperties: PdfExportProperties = {
      hierarchyExportMode: "Expanded"
    };
    grid.pdfExport(exportProperties);
  }
}
function exportDetailTemplate(args: ExportDetailTemplateEventArgs) {
  args.value = {
    columnCount: 2,
    columnHeader: [
      {
        cells: [{
          index: 0, colSpan: 2, value: 'Product Details',
          style: { backgroundColor: '#ADD8E6', pdfTextAlignment:
'Center', bold: true }
        }]
      }
    ],
    rows: [
      {
        cells: [
          {
            index: 0, rowspan: 4, image: { base64:
args.parentRow.data['ProductImg'], width: 80 }
          },
          {
            index: 1, value: "Offers: " +
args.parentRow.data['Offers'],
            style: { fontColor: '#0A76FF', fontSize: 15 }
          },
        ]
      },
    ],
  },
}
```

```

        {
            cells: [
                {
                    index: 1, value: 'Available: ' +
args.parentRow.data['Available']
                }
            ],
        },
        {
            cells: [
                {
                    index: 1, value: 'Contact: ',
                    hyperlink: {
                        target: 'mailto:' +
args.parentRow.data['Contact'],
                        displayText: args.parentRow.data['Contact']
                    }
                }
            ]
        },
        {
            cells: [
                {
                    index: 1, value: 'Ratings: ' +
args.parentRow.data['Ratings'],
                    style: { fontColor: '#0A76FF', fontSize: 15 }
                }
            ]
        },
        {
            cells: [
                {
                    index: 0, value: args.parentRow.data['productDesc'],
                    style: { pdfTextAlignment: 'Center' }
                },
                { index: 1, value: args.parentRow.data['ReturnPolicy'] }
            ]
        },
        {
            cells: [
                {
                    index: 0, value: args.parentRow.data['Cost'],
                    style: { bold: true, pdfTextAlignment: 'Center' }
                },
                { index: 1, value: args.parentRow.data['Cancellation'] }
            ]
        },
        {
            cells: [
                {
                    index: 0, value: args.parentRow.data['Status'],
                    style: {
                        fontColor: args.parentRow.data['Status'] ===
'Available' ? '#00FF00' : '#FF0000',
                        pdfTextAlignment: 'Center', fontSize: 15
                    }
                },
                {
                    index: 1, value: args.parentRow.data['Delivery'],

```

```

        style: { fontColor: '#0A76FF', fontSize: 15 }
    }
    ]
}
],
};
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="detailtemplate">
    <table class="detailtable" width="100%">
      <colgroup>
        <col width="40%" />
        <col width="60%" />

```

```

        </colgroup>
        <thead>
            <tr>
                <th colspan="2" style="font-weight: 500;text-align:
center;background-color: #ADD8E6;">
                    Product Details
                </th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td rowspan="4" style="text-align: center;">
                    
                </td>
                <td>
                    <span style="font-weight: 500;color:
#0a76ff;">Offers: {Offers} </span>
                </td>
            </tr>
            <tr>
                <td>
                    <span>Available: {Available} </span>
                </td>
            </tr>
            <tr>
                <td>
                    <span class="link">
                        Contact: <a
href="mailto:{Contact}">{Contact}</a>
                    </span>
                </td>
            </tr>
            <tr>
                <td>
                    <span style="font-weight: 500;color: #0a76ff;">
Ratings: {Ratings}</span>
                </td>
            </tr>
            <tr>
                <td style="text-align: center;">
                    <span> {productDesc}</span>
                </td>
                <td>
                    <span>{ReturnPolicy}</span>
                </td>
            </tr>
            <tr>
                <td style="text-align: center;">
                    <span style="font-weight: 500;" > {Cost}</span>
                </td>
                <td>
                    <span>{Cancellation}</span>
                </td>
            </tr>
            <tr>
                <td style="text-align: center;">

```

```

        <span class="{Status}" style="font-weight: 500;" >
    ${Status}</span>
        </td>
        <td>
            <span style="font-weight: 500;color:
    #0a76ff;">${Delivery}</span>
        </td>
    </tr>
</tbody>
</table>
</script>
<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

PDF Export			
	Category	Product ID	Status
▶	Suits/Slim	EJ-SU-01	Available
▶	Suits/Classic	EJ-SU-02	Available
▶	Suits/Formal	EJ-SU-03	Available
▶	Phants/Slim	EJ-PH-01	Available
▶	Phants/Classic	EJ-PH-02	Available
▶	Shirts/Slim	EJ-SH-01	Available
▶	Shirts/Formal	EJ-SH-02	Available

Exporting with caption template

The PDF export feature enables exporting of Grid with a caption template to an PDF document.

In the following sample, the customized caption text is exported to PDF using [captionText](#) property in the [exportGroupCaption](#) event.

INDEX.TS

```
import { Grid, Group, Toolbar, PdfExport, ExportGroupCaptionEventArgs } from '@syncfusion/ej2-grids';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { employeeData } from './datasource.ts';
Grid.Inject(Group, Toolbar, PdfExport);
let grid: Grid = new Grid({
  dataSource: employeeData,
  allowGrouping: true,
  groupSettings: { captionTemplate: '#captiontemplate', columns:
  ['EmployeeID'] },
  allowPdfExport: true,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', width: 120 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'City', headerText: 'City' },
    { field: 'Title', headerText: 'Title', width: 170 }
  ],
  toolbarClick: toolbarClick,
  exportGroupCaption: exportGroupCaption,
  height: 273
});
grid.appendTo('#Grid');
function toolbarClick(args: ClickEventArgs) {
  if (args.item.id === 'Grid_pdfexport') {
    grid.pdfExport();
  }
}

function exportGroupCaption(args: ExportGroupCaptionEventArgs) {
  args.captionText = args.data['field'] + ' - ' + args.data['key'];
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
```

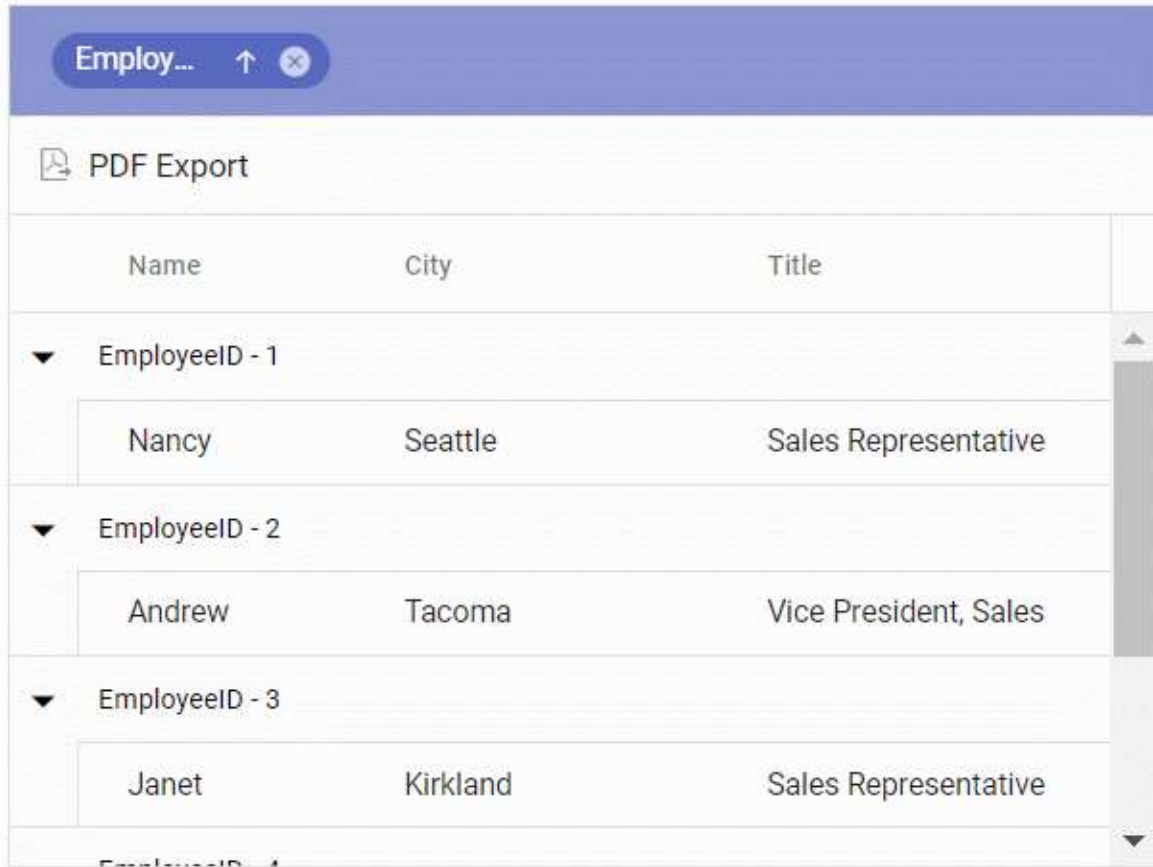
```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="captiontemplate" type="text/x-template">
            ${field} - ${key}
        </script>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Name	City	Title
EmployeeID - 1		
Nancy	Seattle	Sales Representative
EmployeeID - 2		
Andrew	Tacoma	Vice President, Sales
EmployeeID - 3		
Janet	Kirkland	Sales Representative

Exporting grid in server in EJ2 JavaScript Grid control

The Grid have an option to export the data to PDF in server side using Grid server export library.

Server dependencies

The Server side export functionality is shipped in the Syncfusion.EJ2.GridExport package, which is available in Essential Studio and [nuget.org](https://www.nuget.org/packages/Syncfusion.EJ2.GridExport). The following list of dependencies is required for Grid server side PDF exporting action.

- Syncfusion.EJ2
- Syncfusion.EJ2.GridExport
- Syncfusion.Compression.Base
- Syncfusion.Pdf.Base

Server configuration

The following code snippets shows server configuration using ASP.NET MVC Controller Action.

To Export the Grid in server side, You need to call the

[serverPdfExport](#) method for passing the Grid properties to server exporting action.

```
`ts
public ActionResult PdfExport(string gridModel)
{
```

```

GridPdfExport exp = new GridPdfExport();
Grid gridProperty = ConvertGridObject(gridModel);
return exp.PdfExport<OrdersDetails>(gridProperty, OrdersDetails.GetAllRecords());
}

private Grid ConvertGridObject(string gridProperty)
{
    Grid GridModel = (Grid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty, typeof(Grid));
    GridColumnModel cols =
    (GridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
    typeof(GridColumnModel));
    GridModel.Columns = cols.columns;
    return GridModel;
}

public class GridColumnModel
{
    public List<GridColumn> columns { get; set; }
}

public ActionResult DataSource(DataManager dm)
{
    var DataSource = OrderRepository.GetAllRecords();
    DataResult result = new DataResult();
    result.result = DataSource.Skip(dm.Skip).Take(dm.Take).ToList();
    result.count = result.result.Count;
    return Json(result, JsonRequestBehavior.AllowGet);
}
`ts

import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Toolbar);
let data: DataManager = new DataManager({
    url: "Home/DataSource",
    adaptor: new UrlAdaptor
});

```

```

let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['PdfExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
  ],
  height: 265
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    grid.serverPdfExport("Home/PdfExport");
  }
}

```

Note: Refer to the GitHub sample for quick implementation and testing from [here](#).

Export grid as memory stream

The Grid offers an option to export the data as a memory stream instead of downloading it as a file in the browser. To obtain the memory stream of the exported grid, set the **AsMemoryStream** parameter to **true** in the [PdfExport](#) method.

The following code demonstrates how to get the memory stream of exported grid.

```

`ts
public object PdfExport(string gridModel)
{
  GridPdfExport exp = new GridPdfExport();
  Grid gridProperty = ConvertGridObject(gridModel);
  // pass third parameter as true to get the Memory Stream of exported grid data
  return (MemoryStream)exp.PdfExport<OrdersDetails>(gridProperty, OrdersDetails.GetAllRecords(),
  true);
}

```

Merge grid's memory stream

The [Essential PDF](#) library is used to merge multiple memory streams into a single stream. To learn more about the merge option, please refer to this [documentation](#).

You can merge a memory stream, a file stream, and a local file with the Grid's memory stream in the following ways:

Merging with an existing memory stream

If you already have a memory stream, you can directly use it to merge with the Grid's memory stream.

In the following code, the [Merge](#) method of the [PdfDocumentBase](#) class is used to merge the grid's memory stream with an existing memory stream.

```
`ts
using Syncfusion.Pdf;

public MemoryStream ms1; // defines existing memory stream
public object PdfExport(string gridModel)
{
    GridPdfExport exp = new GridPdfExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    // get the memory stream of exported grid data
    MemoryStream ms2 = (MemoryStream)exp.PdfExport<OrdersDetails>(gridProperty,
    OrdersDetails.GetAllRecords(), true);
    //Creates a PDF document.
    PdfDocument finalDoc = new PdfDocument();
    //Creates a PDF stream for merging. ms1 and ms2 represents the existing stream and grid's stream.
    Stream[] streams = { ms1, ms2 };
    //Merges PDFDocument.
    PdfDocumentBase.Merge(finalDoc, streams);
    //Save the document into stream.
    MemoryStream ms3 = new MemoryStream();
    finalDoc.Save(ms3);
    ms3.Position = 0;
    //Close the document.
    finalDoc.Close(true);
    //Dispose the streams.
    ms1.Dispose();
    ms2.Dispose();
    return ms3;
}
```

```
}
`ts
```

Merging with an existing file stream

If you already have a file stream, you can directly use it to merge with the Grid's memory stream. In the following code, the existing file stream is merged with the Grid's memory stream.

```
`ts
using Syncfusion.Pdf;

public FileStream fs1; // defines existing file stream

public ActionResult PdfExport(string gridModel)
{
    GridPdfExport exp = new GridPdfExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    MemoryStream ms1 = (MemoryStream)exp.PdfExport<OrdersDetails>(gridProperty,
    OrdersDetails.GetAllRecords(), true);
    PdfDocument finalDoc = new PdfDocument();
    //fs1 and ms1 represents the existing stream and grid's stream.
    Stream[] streams = { fs1, ms1 };
    PdfDocumentBase.Merge(finalDoc, streams);
    MemoryStream ms3 = new MemoryStream();
    finalDoc.Save(ms3);
    ms3.Position = 0;
    return ms3;
}
`ts
```

Merging with a local file

To merge a local file with the Grid's memory stream, you need to convert it into a file stream before merging. In the following code, the existing local file is merged with the Grid's memory stream.

```
`ts
using Syncfusion.Pdf;

// get the file stream of local file

public FileStream fs1 = new FileStream("D:/PdfDoc.pdf", FileMode.Open, FileAccess.Read); //
PdfDoc.pdf is a local file which is located in local disk D.

public ActionResult PdfExport(string gridModel)
{
    GridPdfExport exp = new GridPdfExport();
```

```

Grid gridProperty = ConvertGridObject(gridModel);
MemoryStream ms1 = (MemoryStream)exp.PdfExport<OrdersDetails>(gridProperty,
OrdersDetails.GetAllRecords(), true);
PdfDocument finalDoc = new PdfDocument();
//fs1 and ms1 represents the local file's stream and grid's stream.
Stream[] streams = { fs1, ms1 };
PdfDocumentBase.Merge(finalDoc, streams);
MemoryStream ms3 = new MemoryStream();
finalDoc.Save(ms3);
ms3.Position = 0;
return ms3;
}
`

```

Downloading the merged memory stream

You can download the merged memory stream by converting it into a `FileStreamResult`. In the following code, the merged memory stream is downloaded to the browser.

```

`ts
using Syncfusion.Pdf;
public ActionResult PdfExport(string gridModel)
{
    PdfDocument finalDoc = new PdfDocument();
    //ms1 and ms2 represents the streams needs to merge.
    Stream[] streams = { ms1, ms2 };
    PdfDocumentBase.Merge(finalDoc, streams);
    MemoryStream ms3 = new MemoryStream();
    finalDoc.Save(ms3);
    ms3.Position = 0;
    // Save the MemoryStream into FileStreamResult
    FileStreamResult fileStreamResult = new FileStreamResult(ms3, "application/pdf");
    fileStreamResult.FileName = "Export.pdf";
    //Close the document.
    finalDoc.Close(true);
    //Dispose the streams.
    ms1.Dispose();
}
`

```

```

ms2.Dispose();
// return the file
return fileStreamResult;
}
`

```

Rotate a header text to a certain degree in the exported grid on the server side

The Grid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported PDF file. To achieve this requirement, define the `BeginCellLayout` event of the `PdfExportProperties` with an event handler to perform the required action.

The `PdfHeaderQueryCellInfoEvent` will be triggered when creating a column header for the pdf document to be exported. Collect the column header details in this event and handle the custom in the `BeginCellLayout` event handler.

In the following demo, the `DrawString` method from the `Graphics` is used to rotate the header text of the column header inside the `BeginCellLayout` event handler.

A PDF exporting is not supported to rotate the column header on the client side.

```

`ts
public ActionResult PdfExport(string gridModel)
{
    GridPdfExport exp = new GridPdfExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    gridProperty.ServerPdfHeaderQueryCellInfo = PdfHeaderQueryCellInfo;
    PdfGrid grid = new PdfGrid();
    PdfExportProperties pdfExportProperties = new PdfExportProperties();
    pdfExportProperties.IsRepeatHeader = true;
    pdfExportProperties.BeginCellLayout = new PdfGridBeginCellLayoutEventHandler(BeginCellEvent);
    gridProperty.ServerPdfQueryCellInfo = PdfQueryCellInfo;
    IEnumerable data = Utils.DataTableToJson(dt);
    var result = exp.PdfExport<dynamic>(gridProperty, data, pdfExportProperties);
    return result;
}

public void BeginCellEvent(object sender, PdfGridBeginCellLayoutEventArgs args)
{
    PdfGrid grid = (PdfGrid)sender;
    var brush = new PdfSolidBrush(new PdfColor(Color.DimGray));
    args.Graphics.Save();
}

```

```

args.Graphics.TranslateTransform(args.Bounds.X + 50, args.Bounds.Height + 40); // give the value for
bounds x and Y by the user

args.Graphics.RotateTransform(-60); // give the rotate degree value by the user

// Draw the text at particular bounds.

args.Graphics.DrawString(headerValues[args.CellIndex], new PdfStandardFont(PdfFontFamily.Helvetica,
10), brush, new PointF(0, 0));

if (args.IsHeaderRow)
{
grid.Headers[0].Cells[args.CellIndex].Value = string.Empty;
}

args.Graphics.Restore();
}

private void PdfHeaderQueryCellInfo(object pdf)
{
ServerPdfHeaderQueryCellInfoEventArgs name = (ServerPdfHeaderQueryCellInfoEventArgs)pdf;
PdfGrid grid = new PdfGrid();
headerValues.Add(name.Column.HeaderText);
var longestString = headerValues.Where(s => s.Length == headerValues.Max(m => m.Length)).First();
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 6);
SizeF size = font.MeasureString(longestString);
name.Headers[0].Height = size.Width * 2;
}

```

Limitations

- The export feature for detail templates is not supported in server-side exporting.
- Multiple grids exporting feature is not supported with server side exporting.

Excel Export

Excel exporting in EJ2 JavaScript Grid control

The excel export allows exporting Grid data to Excel document. You need to use the [excelExport](#) method for exporting. To enable Excel export in the grid, set the [allowExcelExport](#) as true.

To use excel export, You need to inject the **ExcelExport** module in grid.

INDEX.TS

```

import { Grid, Toolbar, ExcelExport, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page);

```



```

let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],
    height: 230
});
grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_excelexport') {
        grid.excelExport();
    }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show spinner while exporting

You can show/ hide spinner component while exporting the grid using `showSpinner/ hideSpinner` methods. You can use `toolbarClick` event to show spinner before exporting and hide a spinner in the `pdfExportComplete` or `excelExportComplete` event after the exporting.

In the `toolbarClick` event, based on the parameter `args.item.id` as `Gridpdfexport` or `Gridexcelexport` we can call the `showSpinner` method from grid instance.

In the `pdfExportComplete` or `excelExportComplete` event, We can call the `hideSpinner` method.

In the below demo, we have rendered the default spinner component when exporting the grid.

INDEX.TS

```

import { Grid, Page, Toolbar, PdfExport, ExcelExport } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, PdfExport, ExcelExport);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  allowExcelExport: true,
  toolbar: ['PdfExport', 'ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID',
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
width: 140 }
  ],
  height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    grid.showSpinner();
    grid.pdfExport();
  }
  if (args['item'].id === 'Grid_excelexport') {
    grid.showSpinner();
    grid.excelExport();
  }
}
grid.pdfExportComplete = () => {
  grid.hideSpinner();
}
grid.excelExportComplete = () => {
  grid.hideSpinner();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom data source

The excel export provides an option to define datasource dynamically before exporting. To export data dynamically, define the `dataSource` in `exportProperties`.

INDEX.TS

```
import { Grid, Toolbar, ExcelExport, ExcelExportProperties, Page } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
  ],
  height: 230
});
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_excelexport') {
    let excelExportProperties: ExcelExportProperties = {
      dataSource: data
    };
    grid.excelExport(excelExportProperties);
  }
}
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Passing additional parameters to the server when exporting

You can pass the additional parameter in the `query` property by invoking `addParams` method. In the `toolbarClick` event, you can define params as key and value pair so it will receive at the server side when exporting.

In the below example, we have passed `recordcount` as `12` using `addParams` method.

INDEX.TS

```
import { Grid, Page, Toolbar, PdfExport, ExcelExport } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { Query } from '@syncfusion/ej2-data';
Grid.Inject(Page, Toolbar, PdfExport, ExcelExport);
let queryClone: Object;
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowPdfExport: true,
  allowExcelExport: true,
  toolbar: ['PdfExport', 'ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID',
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'ShipCity', headerText: 'ShipCity', textAlign: 'Right',
width: 140 }
  ],
  height: 230
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_pdfexport') {
    queryClone = grid.query;
    grid.query = new Query().addParams("recordcount", "12");
    grid.pdfExport();
  }
  if (args['item'].id === 'Grid_excelexport') {
    queryClone = grid.query;
    grid.query = new Query().addParams("recordcount", "12");
    grid.excelExport();
  }
}
grid.pdfExportComplete = () => {
  grid.query = queryClone;
}
grid.excelExportComplete = () => {
  grid.query = queryClone;
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

        <div id="container">
            <div id="Grid"></div>
        </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Passing the selected records to the server using ajax request via custom toolbar button click

You can pass the selected records to the server with the help of an ajax request. In the `toolbarClick` event, you can get the selected records using the [getSelectedRecords](#) method and pass the selected records to the server using the **data** property of the ajax.

`ts

```

import { Grid, Page, Toolbar } from '@syncfusion/ej2-grids';
import { Ajax } from '@syncfusion/ej2-base';

Grid.Inject(Page, Toolbar);

let grid: Grid = new Grid({
    dataSource: data,
    allowSelection: true,
    toolbar: ['Selected'],
    columns: [
        { type: 'checkbox', width: 50, },
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 120},
        { field: 'CustomerID', width: 140, headerText: 'CustomerID'},
        { field: 'ShipCity', headerText: 'Ship Country', width: 140 }
    ],
    toolbarClick: function (args) {
        if (args.item.text === 'Selected') {
            var selectedRecord = this.getSelectedRecords();
            var ajax = new Ajax({
                url: "/Home/SelectRecord",
                type: "POST",
                contentType: "application/json",
                data: JSON.stringify(selectedRecord),
                successHandler: function (response) {

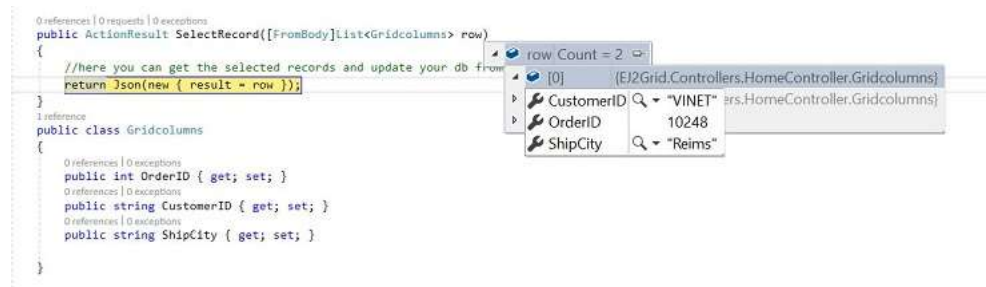
```

```

console.log(JSON.parse(response));
},
failure: function (response) {
alert(response);
}
});
ajax.send();
}
},
});
grid.appendTo('#Grid');
`

```

The selected record details are bound to the `row` parameter. Please refer to the following screenshot.



[See Also](#)

- [Exporting Grid in Cordova application](#)

Export multiple grids in EJ2 JavaScript Grid control

The Excel export provides an option to export multiple grid data in the same or different sheets of an Excel file. Each grid is identified by its unique ID. You can specify which grids to export by listing their IDs in the `exportGrids` property.

Same sheet

Excel exporting provides support for exporting multiple grids on the same sheet. To export the grids in the same sheet, define `multipleExport.type` as **AppendToSheet** in `exportProperties`. It also has an option to provide blank rows between the grids. These blank rows count can be defined by using `multipleExport.blankRows` property.

INDEX.TS

```

import { Grid, Toolbar, ExcelExport, ExcelExportProperties, Page } from
 '@syncfusion/ej2-grids';
import { data, employeeData } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page);
let firstGrid: Grid = new Grid({

```

```

dataSource: data.slice(0, 5),
allowPaging: true,
exportGrids: ['FirstGrid', 'SecondGrid'],
allowExcelExport: true,
toolbar: ['ExcelExport'],
columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
]
});
let secondGrid: Grid = new Grid({
    dataSource: employeeData.slice(0, 5),
    allowPaging: true,
    allowExcelExport: true,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Reight', width: 120, type: 'number' },
        { field: 'FirstName', width: 140, headerText: 'First Name', type:
'string' },
        { field: 'LastName', width: 140, headerText: 'Last Name', type:
'string' },
        { field: 'City', headerText: 'City', width: 120 },
    ]
});
firstGrid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'FirstGrid_excelelexport') {
        let appendExcelExportProperties: ExcelExportProperties = {
            multipleExport: { type: 'AppendToSheet', blankRows: 2 }
        };
        firstGrid.excelExport(appendExcelExportProperties, true);
    }
}
firstGrid.appendTo('#FirstGrid');
secondGrid.appendTo('#SecondGrid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <p><b>First Grid</b></p>
        <div id="FirstGrid"></div>
        <p><b>Second Grid</b></p>
        <div id="SecondGrid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, `multipleExport.blankRows` value is 5.

New sheet

Excel export functionality enables the exporting of multiple grids onto separate sheets (each grid in new sheet of excel) within the Excel file. To achieve this, you can specify `multipleExport.type` as `NewSheet` in `exportProperties`.

INDEX.TS

```

import { Grid, Toolbar, ExcelExport, ExcelExportProperties, Page } from
 '@syncfusion/ej2-grids';
import { data, employeeData } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page);
let firstGrid: Grid = new Grid({

```

```

dataSource: data.slice(0, 5),
allowPaging: true,
allowExcelExport: true,
exportGrids: ['FirstGrid', 'SecondGrid'],
toolbar: ['ExcelExport'],
columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
]
});
let secondGrid: Grid = new Grid({
dataSource: employeeData.slice(0, 5),
allowPaging: true,
allowExcelExport: true,
columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120, type: 'number' },
    { field: 'FirstName', width: 140, headerText: 'First Name', type:
'string' },
    { field: 'LastName', width: 140, headerText: 'Last Name', type:
'string' },
    { field: 'City', headerText: 'City', width: 120 },
]
});
firstGrid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'FirstGrid_excelexport') {
        let appendExcelExportProperties: ExcelExportProperties = {
            multipleExport: { type: 'NewSheet' }
        };
        firstGrid.excelExport(appendExcelExportProperties, true);
    }
}
firstGrid.appendTo('#FirstGrid');
secondGrid.appendTo('#SecondGrid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <p><b>First Grid</b></p>
        <div id="FirstGrid"></div>
        <p><b>Second Grid</b></p>
        <div id="SecondGrid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Excel export options in EJ2 JavaScript Grid control

The excel export provides an option to customize mapping of the grid to excel document.

Export current page

The excel export provides an option to export the current page into excel. To export current page, define `exportType` to `CurrentPage`.

INDEX.TS

```

import { Grid, Toolbar, ExcelExport, ExcelExportProperties, Page } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page);
let grid: Grid = new Grid({

```

```

    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],
    height: 230
});
grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_excelexport') {
        let excelExportProperties: ExcelExportProperties = {
            exportType: 'CurrentPage'
        };
        grid.excelExport(excelExportProperties);
    }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export the selected records only

You can export the selected records data by passing it to `exportProperties.dataSource` Property in the `toolbarClick` event.

In the below exporting demo, We can get the selected records using `getSelectedRecords` method and pass the selected data to `PdfExport` or `excelExport` property.

INDEX.TS

```

import { Grid, Page, Toolbar, PdfExport, Filter, ExcelExport } from
 '@syncfusion/ej2-grids';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';

```



```

import { data } from './datasource.ts';
import { DataManager } from '@syncfusion/ej2-data';
Grid.Inject(Page, Toolbar, PdfExport, Filter, ExcelExport);
let grid: Grid = new Grid(
    {
        dataSource: data,
        allowPdfExport: true,
        allowExcelExport: true,
        allowPaging: true,
        allowFiltering: true,
        selectionSettings: { type: 'Multiple' },
        toolbar: ['PdfExport', 'ExcelExport'],
        pageSettings: { pageCount: 5, pageSize: 5 },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', width: 120,
textAlign: 'Right' },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'OrderDate', headerText: 'Order Date', width: 130,
format: 'yMd', textAlign: 'Right' },
            { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right'
},
            { field: 'ShipCountry', visible: false, headerText: 'Ship
Country', width: 150 }
        ],
    });
grid.appendTo('#Grid');
grid.toolbarClick = (args: ClickEventArgs) => {
    if (args.item.id === 'Grid_pdfexport') {
        let pdfdata = grid.getSelectedRecords();
        let exportProperties = {
            dataSource: pdfdata,
        };
        grid.pdfExport(exportProperties);
    }
    else if (args.item.id === 'Grid_excelexport') {
        let exceldata = grid.getSelectedRecords();
        let exportProperties = {
            dataSource: exceldata,
        };
        grid.excelExport(exportProperties);
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Export hidden columns

The excel export provides an option to export hidden columns of grid by defining `includeHiddenColumn` as `true`.

INDEX.TS

```
import { Grid, Toolbar, ExcelExport, ExcelExportProperties, Page } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page)
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'ShipCountry', width: 140, headerText: 'Ship Country',
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
  ],
  height: 230
});
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_excelexport') {
    let excelExportProperties: ExcelExportProperties = {
      includeHiddenColumn: true
    };
    grid.excelExport(excelExportProperties);
  }
}
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Show or hide columns

You can show a hidden column or hide a visible column while printing the grid using [toolbarClick](#) and [excelExportComplete](#) events.

In the `toolbarClick` event, based on `args.item.id` as `Grid_excelexport`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `excelExportComplete` event, We have reversed the state back to the previous state.

In the below example, we have `CustomerID` as a hidden column in the grid. While exporting, we have changed `CustomerID` to visible column and `ShipCity` as hidden column.

INDEX.TS

```
import { Grid, Toolbar, ExcelExport, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID',
visible: false },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'ShipCity', headerText: 'ShipCity', width: 140, textAlign:
'Right' }
  ],
  height: 230
});
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_excelexport') {
    grid.columns[1].visible = false;
    grid.columns[3].visible = true;
    grid.excelExport();
  }
}
grid.pdfExportComplete = () => {
  grid.columns[1].visible = false;
  grid.columns[3].visible = true;
}
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
```

```

    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export with filter options

The excel export provides an option to export with filter option in excel by defining `enableFilter` as `true`.

It requires the [allowFiltering](#) to be true.

INDEX.TS

```

import { Grid, Toolbar, ExcelExport, ExcelExportProperties, Page, Filter }
from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page, Filter);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    allowFiltering: true,
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],
    height: 230
});
grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_excelexport') {
        let excelExportProperties: ExcelExportProperties = {
            enableFilter: true
        };
        grid.excelExport(excelExportProperties);
    }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');

```



```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting grouped records

The excel export provides outline option for grouped records which hides the detailed data for better viewing. In grid, we have provided the outline option for the exported document when the data's are grouped.

INDEX.TS

```

import { Grid, Toolbar, ExcelExport, Group } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Group);
let grid: Grid = new Grid({
    dataSource: data,
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    allowGrouping: true,
    groupSettings: { columns: ['OrderID', 'CustomerID'] };
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],
    height: 230
});
grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_excelexport') {
        grid.excelExport();
    }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Define file name

You can assign the file name for the exported document by defining `fileName` property in [ExcelExportProperties](#).

INDEX.TS

```
import { Grid, Toolbar, ExcelExport, ExcelExportProperties, Page } from
 '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
  ],
  height: 230
});
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_excelexport') {
    let excelExportProperties: ExcelExportProperties = {
      fileName: "new.xlsx"
    };
    grid.excelExport(excelExportProperties);
  }
}
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export the master detail grid

It is possible to export the master-detail grid on the same Excel sheet using the `ExcelExportProperties` class in the grid.

To export the master-detail grid on the same sheet in the following sample, you need to set the `multipleExport.type` to `AppendToSheet` in the exportProperties. A promise object is created by exporting the master grid first, and then the detail grid is exported after the master grid has been successfully exported.

INDEX.TS

```
import { Grid, RowSelectEventArgs, Selection, Toolbar, ExcelExport } from
 '@syncfusion/ej2-grids';
import { data, customerData } from './datasource.ts';
Grid.Inject(Selection, Toolbar, ExcelExport);
type carType = { CustomerID: string; CustomerName: string; ContactName:
 string; };
let names: string[] = ['AROUT', 'BERGS', 'BLONP', 'CHOPS', 'ERNSH'];
let masterdata: Object = customerData.filter((e: carType) =>
 names.indexOf(e.CustomerID) !== -1);
let mastergrid: Grid = new Grid({
  dataSource: masterdata,
  toolbar: ['ExcelExport'],
  selectedRowIndex: 1,
  allowExcelExport: true,
  toolbarClick: toolbarClick,
  columns: [
    { field: 'ContactName', headerText: 'Customer Name', width: 150 },
    { field: 'CompanyName', headerText: 'Company Name', width: 150 },
    { field: 'Address', headerText: 'Address', width: 150 },
    { field: 'Country', headerText: 'Country', width: 130 },
  ],
  rowSelected: rowSelected,
});
mastergrid.appendTo('#MasterGrid');
function rowSelected(args: RowSelectEventArgs): void {
  let selectedRecord: carType = args.data as carType;
  grid.dataSource = data.filter((record: carType) => record.CustomerName
  === selectedRecord.ContactName).slice(0, 5);
  document.getElementById('key').innerHTML = selectedRecord.ContactName;
}
function toolbarClick(args) {
  if (args.item.id === 'MasterGrid_excelexport') {
    const appendExcelExportProperties: ExcelExportProperties = {
      multipleExport: { type: 'AppendToSheet', blankRows: 2 },
    };
    const firstGridExport: Promise<any> =
mastergrid.excelExport(appendExcelExportProperties, true);
    firstGridExport.then((fData: any) => {
      grid.excelExport(appendExcelExportProperties, false, fData);
    });
  }
}
let grid: Grid = new Grid({
  allowSelection: false,
```

```

        allowExcelExport: true,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', width: 100, textAlign:
'Right' },
            { field: 'Freight', headerText: 'Freight', width: 100, format: 'C2',
type: 'number' },
            { field: 'ShipName', headerText: 'Ship Name', width: 200 },
            { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
            { field: 'ShipAddress', headerText: 'Ship Address', width: 200 },
        ],
    });
    grid.appendTo('#DetailGrid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <p class="e-mastertext">Master Grid</p>
      <div id="MasterGrid">
      </div>
      <p class="e-mastertext">Detail Grid</p>
      <div id="DetailGrid">
      </div>
    </p></div>
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Export grid as blob

The Grid offers an option to export the data as a **Blob** instead of downloading it as a file in the browser. To export the grid as a Blob, set the **isBlob** parameter to **true** in the [excelExport](#) method. The grid returns the promise of a blob in the [excelExportComplete](#) event.

The following example demonstrates how to obtain the blob data of the exported grid by executing the promise in the **excelExportComplete** event.

INDEX.TS

```

import { Grid, Page, Toolbar, ExcelExport, ExcelExportCompleteArgs } from
 '@syncfusion/ej2-grids';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, ExcelExport);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowExcelExport: true,
  toolbar: ['ExcelExport', 'CsvExport'],
  toolbarClick: toolbarClick,
  excelExportComplete: excelExportComplete,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  height: 230
});
grid.appendTo('#Grid');
function toolbarClick(args: ClickEventArgs) {
  if (args.item.id === 'Grid_excelexport') {

```

```

        // pass fourth parameter as true to get the blob data of exported
grid
        grid.excelExport(null, null, null, true);
    }
    if (args.item.id === 'Grid_csvexport') {
        // pass fourth parameter as true to get the blob data of exported
grid
        grid.csvExport(null, null, null, true);
    }
}
function excelExportComplete(args: ExcelExportCompleteArgs) {
    // execute the promise to get the blob data
    args.promise.then((e: { blobData: Blob }) => {
        console.log(e.blobData);
    });
};
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```



```
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Excel cell style customization in EJ2 JavaScript Grid control

Conditional cell formatting

Grid cells in the exported Excel can be customized or formatted using [excelQueryCellInfo](#) event. In this event, we can format the grid cells of exported PDF document based on the column cell value.

In the below sample, we have set the background color for **Freight** column in the exported excel by **args.cell** and **backColor** property.

INDEX.TS

```
import { Grid, Toolbar, ExcelExport, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
  ],
  height: 230
});
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_excelexport') {
    grid.excelExport();
  }
}
grid.excelQueryCellInfo = (args: Object) => {
  if (args.column.field === 'Freight') {
    if (args.value < 30) {
      args.style = { backColor: '#99ffcc' };
    }
    else if (args.value < 60) {
```

```

        args.style = {backColor: '#ffffb3'};
    }
    else {
        args.style = {backColor: '#ff704d'};
    }
}
}
grid.queryCellInfo = (args: Object) => {
    if(args.column.field == 'Freight'){
        if(args.data['Freight'] < 30) {
            args.cell.bgColor = '#99ffcc';
        }
        else if(args.data['Freight'] < 60) {
            args.cell.bgColor = '#ffffb3';
        }
        else {
            args.cell.bgColor = '#ff704d';
        }
    }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Theme

The excel export provides an option to include theme for exported excel document.

To apply theme in exported Excel, define the **theme** in **exportProperties**.

INDEX.TS

```

import { Grid, Toolbar, ExcelExport, ExcelExportProperties, Page } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    columns: [

```

```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],
    height: 230
});
grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_excelexport') {
        let excelExportProperties: ExcelExportProperties = {
            theme:
            {
                header: { fontName: 'Segoe UI', fontColor: '#666666' },
                record: { fontName: 'Segoe UI', fontColor: '#666666' },
                caption: { fontName: 'Segoe UI', fontColor: '#666666' }
            }
        };
        grid.excelExport(excelExportProperties);
    }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, material theme is applied to exported excel document.

Rotate a header text to a certain degree in the exported grid

The DataGrid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported Excel file. To achieve this requirement, use the [excelHeaderQueryCellInfo](#) event of the Grid.

The `excelHeaderQueryCellInfo` will be triggered when creating a column header for the excel document to be exported. Customize the column header in this event.

In the following demo, using the `rotation` property of the style argument in the `excelHeaderQueryCellInfo` event, you can rotate the header text of the column header in the excel exported document.

INDEX.TS

```
import { Grid, Toolbar, ExcelExport, ExcelExportProperties, Page,
  ActionEventArgs } from '@syncfusion/ej2-grids';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page);
let grid: Grid = new Grid({
  dataSource: data,
  allowPaging: true,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string', customAttributes:{ class: 'orientationcss' }, textAlign: 'Center'
},
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120 },
    { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
  ],
  created: setHeaderHeight,
  excelHeaderQueryCellInfo: (args) => {
    let textWidth = document.querySelector(".orientationcss >
div").scrollWidth;
    if (args.gridCell.column.field == 'Freight') {
      args.style = { backgroundColor: '#99ffcc', vAlign: 'Bottom' };
    }
    else {
      args.style = { vAlign: 'Center', rotation:
dropDownListObject.value };
    }
    args.cell.cellHeight = textWidth;
  },
  excelQueryCellInfo: (args) => {
    if (args.column.field == 'Freight') {
      if (args.value < 30) {
        args.style = { backgroundColor: '#99ffcc' };
      }
      else if (args.value < 60) {
        args.style = { backgroundColor: '#ffffb3' };
      }
      else {
        args.style = { backgroundColor: '#ff704d' };
      }
    }
  }
  height: 230
});
grid.appendTo('#Grid');
function setHeaderHeight(args: ActionEventArgs): void {
```

```

    let textWidth: number = document.querySelector(".orientationcss >
div").scrollWidth;
    let headerCell: NodeList = document.querySelectorAll(".e-headercell");
    for (let i: number = 0; i < headerCell.length; i++) {
        (<HTMLElement>headerCell.item(i)).style.height = textWidth + 'px';
    }
}
grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_excelexport') {
        grid.excelExport();
    }
}
let degree = [90, 180, 45, 135, 225, -90];
let dropDownListObject: DropDownList = new DropDownList({
    dataSource: degree,
    placeholder: "Select a degree"
});
dropDownListObject.appendTo('#ddlelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="toolbar-template">
            <div id="dropdown" style="margin-top:5px">
                <input type="text" tabindex="1" id="ddlelement">
            </div>
        </div>
        <div id="Grid"></div>
    </div>
<style>
    .orientationcss .e-headercelldiv {
        transform: rotate(90deg);
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding header and footer in EJ2 JavaScript Grid control

The excel export provides an option to include header and footer content for exported excel document.

INDEX.TS

```

import { Grid, Toolbar, ExcelExport, ExcelExportProperties, Page } from
'@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport, Page);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowExcelExport: true,
    toolbar: ['ExcelExport'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd', textAlign: 'Right' }
    ],
    height: 230
});

```



```

grid.toolbarClick = (args: Object) => {
    if (args['item'].id === 'Grid_excelexport') {
        let excelExportProperties: ExcelExportProperties = {
            header: {
                headerRows: 7,
                rows: [
                    { cells: [{ colSpan: 4, value: "Northwind Traders",
style: { fontColor: '#C67878', fontSize: 20, hAlign: 'Center', bold: true, }
}] },
                    { cells: [{ colSpan: 4, value: "2501 Aerial Center
Parkway", style: { fontColor: '#C67878', fontSize: 15, hAlign: 'Center',
bold: true, } } ] },
                    { cells: [{ colSpan: 4, value: "Suite 200 Morrisville,
NC 27560 USA", style: { fontColor: '#C67878', fontSize: 15, hAlign:
'Center', bold: true, } } ] },
                    { cells: [{ colSpan: 4, value: "Tel +1 888.936.8638 Fax
+1 919.573.0306", style: { fontColor: '#C67878', fontSize: 15, hAlign:
'Center', bold: true, } } ] },
                    { cells: [{ colSpan: 4, hyperlink: { target:
'https://www.northwind.com/', displayText: 'www.northwind.com' }, style: {
hAlign: 'Center' } } ] },
                    { cells: [{ colSpan: 4, hyperlink: { target:
'mailto:support@northwind.com' }, style: { hAlign: 'Center' } } ] },
                ]
            },
            footer: {
                footerRows: 4,
                rows: [
                    { cells: [{ colSpan: 4, value: "Thank you for your
business!", style: { hAlign: 'Center', bold: true } } ] },
                    { cells: [{ colSpan: 4, value: "!Visit Again!", style: {
hAlign: 'Center', bold: true } } ] }
                ]
            },
        };
        grid.excelExport(excelExportProperties);
    }
}
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting hierarchy grid in EJ2 JavaScript Grid control

The grid has an option to export the hierarchy grid to excel document. By default, it will use **Expanded** as hierarchyExportMode. you can change the exporting option by using the **ExcelExportProperties.hierarchyExportMode** property. The available options are,

| Mode | Behavior |

|-----|-----|

| Expanded | Exports the visible child grids in expanded state and remaining child grid in collapsed state when args.isChild property is set to true in [beforeExcelExport](#) event. |

| All | Exports the all the child grids in expanded state. |

| None | Exports all child grids in collapsed state when args.isChild property is set to true in [beforeExcelExport](#) event. |

INDEX.TS

```
import { Grid, DetailRow, Toolbar, ExcelExport, ExcelExportProperties } from
 '@syncfusion/ej2-grids';
import { data, employeeData } from './datasource.ts';
Grid.Inject(DetailRow, Toolbar, ExcelExport);
let grid: Grid = new Grid({
  dataSource: employeeData,
  toolbar: ['ExcelExport'],
  allowExcelExport: true,
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
    { field: 'FirstName', headerText: 'First Name', width: 150 },
    { field: 'City', headerText: 'City', width: 150 },
    { field: 'Country', headerText: 'Country', width: 150 }
  ],
  childGrid: {
    dataSource: data,
    queryString: 'EmployeeID',
    columns: [
      { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
      { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
      { field: 'ShipCity', headerText: 'Ship City', width: 150 },
      { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
  },
  beforeExcelExport: beforeExcelExport,
  toolbarClick: toolbarClick
});
grid.appendTo('#Grid');
function beforeExcelExport(args) {
  args.isChild = true;
}
function toolbarClick(args) {
  if (args['item'].id === 'Grid_excelexport') {
    let exportProperties: ExcelExportProperties = {
      hierarchyExportMode: "Expanded"
    }
  }
}
```

```

    };
    grid.excelExport (exportProperties);
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations

- Microsoft Excel permits up to seven nested levels in outlines. So that in the grid we can able to provide only up to seven nested level

and if it exceeds more than seven levels then the document will be exported without outline option. Please refer the [Microsoft Limitation](#)

Exporting grid with templates in EJ2 JavaScript Grid control

The grid offers the option to export the column, detail, and caption templates to an Excel document. The template contains images, hyperlinks, and customized text.

Exporting with column template

The Excel export functionality allows you to export Grid columns that include images, hyperlinks, and custom text to an Excel document.

In the following sample, the hyperlinks and images are exported to Excel using [hyperlink](#) and [image](#) properties in the [excelQueryCellInfo](#) event.

Excel Export supports base64 string to export the images.

INDEX.TS

```

import { Grid, Toolbar, ExcelExport, ExcelQueryCellInfoEventArgs } from
 '@syncfusion/ej2-grids';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { employeeData } from './datasource.ts';
Grid.Inject(Toolbar, ExcelExport);
let grid: Grid = new Grid({
  dataSource: employeeData,
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    {
      headerText: 'Employee Image', textAlign: 'Center',
      template: '#imageTemplate', width: 150
    },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 125 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'EmailID', headerText: 'Email ID', template:
'#mailTemplate', width: 170 }
  ],
  toolbarClick: toolbarClick,
  excelQueryCellInfo: excelQueryCellInfo,
  height: 273
});
grid.appendTo('#Grid');
function toolbarClick(args: ClickEventArgs) {
  if (args.item.id === 'Grid_excelExport') {

```

```

        grid.excelExport();
    }
}
function excelQueryCellInfo(args: ExcelQueryCellInfoEventArgs): any {
    if (args.column.headerText === 'Employee Image') {
        args.image = {
            base64: args.data['EmployeeImage'],
            height: 50,
            width: 50,
        };
    }
    if (args.column.headerText === 'Email ID') {
        args.hyperLink = {
            target: 'mailto:' + args.data['EmailID'],
            displayText: args.data['EmailID'],
        };
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">





    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="imageTemplate" type="text/x-template">
            <div class="image">
                
            </div>
        </script>
        <script id="mailTemplate" type="text/x-template">
            <div class="link">
                <a href="mailto:{EmailID}">{EmailID}</a></div>
            </div>
        </script>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Excel Export		
Employee Image	Name	Email ID
	Nancy	nancy@domain.com
	Andrew	andrew@domain.com
	Janet	janet@domain.com
	Margaret	margaret@domain.com

Exporting with detail template

By default, the grid will export the parent grid with expanded detail rows alone. Change the exporting option by using the `ExcelExportProperties.hierarchyExportMode` property. The available options are:

- | Mode | Behavior |
- |-----|-----|
- | Expanded | Exports the parent grid with expanded detail rows. |
- | All | Exports the parent grid with all the detail rows. |
- | None | Exports the parent grid alone. |

The detail rows in the exported Excel can be customized or formatted using the [exportDetailTemplate](#) event. In this event, the detail rows of the Excel document are formatted in accordance with their parent row details.

In the following sample, the detail row content is formatted by specifying the [columnHeader](#) and [rows](#) properties using its [parentRow](#) details. This allows for the creation of detail rows in the Excel document. Additionally, custom styles can be applied to specific cells using the [style](#) property.

When using [rowSpan](#), it is essential to provide the cell's [index](#) for proper functionality.

INDEX.TS


```

import { Grid, DetailRow, Toolbar, ExcelExport, ExcelExportProperties,
ExportDetailTemplateEventArgs } from '@syncfusion/ej2-grids';
import { employeeData } from './datasource.ts';
Grid.Inject(DetailRow, Toolbar, ExcelExport);
let grid: Grid = new Grid({
    dataSource: employeeData,
    detailTemplate: '#detailtemplate',
    toolbar: ['ExcelExport'],
    allowExcelExport: true,
    toolbarClick: toolbarClick,
    exportDetailTemplate: exportDetailTemplate,
    columns: [
        { field: 'Category', headerText: 'Category', width: 120 },
        { field: 'ProductID', headerText: 'Product ID', width: 140 },
        { field: 'Status', headerText: 'Status', width: 200 }
    ],
    height: 273
});
grid.appendTo('#Grid');
function toolbarClick(args) {
    if (args['item'].id === 'Grid_excelexport') {
        let exportProperties: ExcelExportProperties = {
            hierarchyExportMode: "Expanded"
        };
        grid.excelExport(exportProperties);
    }
}
function exportDetailTemplate(args: ExportDetailTemplateEventArgs) {
    args.value = {
        columnHeader: [
            {
                cells: [{
                    index: 0, colSpan: 2, value: 'Product Details',
                    style: { backgroundColor: '#ADD8E6', excelHAlign: 'Center',
bold: true }
                }]
            }
        ],
        rows: [
            {
                cells: [
                    {
                        index: 0, rowspan: 4, image: {
                            base64: args.parentRow.data['ProductImg'],
                            height: 80, width: 100
                        }
                    },
                    {
                        index: 1, value: "Offers: " +
args.parentRow.data['Offers'],
                        style: { bold: true, fontColor: '#0a76ff' }
                    },
                ]
            },
            {
                cells: [
                    {

```

```

        index: 1, value: 'Available: ' +
args.parentRow.data['Available']
    ]],
    },
    {
        cells: [
            {
                index: 1, value: 'Contact: ',
                hyperlink: {
                    target: 'mailto:' +
args.parentRow.data['Contact'],
                    displayText: args.parentRow.data['Contact']
                }
            }
        ]
    },
    {
        cells: [
            {
                index: 1, value: 'Ratings: ' +
args.parentRow.data['Ratings'],
                style: { bold: true, fontColor: '#0a76ff' }
            }
        ]
    },
    {
        cells: [
            {
                index: 0, value: args.parentRow.data['productDesc'],
                style: { excelHAlign: 'Center' }
            },
            { index: 1, value: args.parentRow.data['ReturnPolicy'] }
        ]
    },
    {
        cells: [
            {
                index: 0, value: args.parentRow.data['Cost'],
                style: { excelHAlign: 'Center', bold: true }
            },
            { index: 1, value: args.parentRow.data['Cancellation'] }
        ]
    },
    {
        cells: [
            {
                index: 0, value: args.parentRow.data['Status'],
                style: {
                    bold: true, fontColor:
args.parentRow.data['Status'] === 'Available' ? '#00FF00' : '#FF0000',
                    excelHAlign: 'Center'
                }
            },
            {
                index: 1, value: args.parentRow.data['Delivery'],
                style: { bold: true, fontColor: '#0a76ff' }
            }
        ]
    }
]

```

$$\left\{ \begin{array}{l}] , \\ } ; \\ \end{array} \right\}$$

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="detailtemplate">
    <table class="detailtable" width="100%">
      <colgroup>
        <col width="40%" />
        <col width="60%" />
      </colgroup>
      <thead>
        <tr>
```

```

        <th colspan="2" style="font-weight: 500;text-align:
center;background-color: #ADD8E6;">
            Product Details
        </th>
    </tr>
</thead>
<tbody>
    <tr>
        <td rowspan="4" style="text-align: center;">
            
        </td>
        <td>
            <span style="font-weight: 500;color:
#0a76ff;">Offers: {Offers} </span>
        </td>
    </tr>
    <tr>
        <td>
            <span>Available: {Available} </span>
        </td>
    </tr>
    <tr>
        <td>
            <span class="link">
                Contact: <a
href="mailto:{Contact}">{Contact}</a>
            </span>
        </td>
    </tr>
    <tr>
        <td>
            <span style="font-weight: 500;color: #0a76ff;">
Ratings: {Ratings}</span>
        </td>
    </tr>
    <tr>
        <td style="text-align: center;">
            <span> {productDesc}</span>
        </td>
        <td>
            <span>{ReturnPolicy}</span>
        </td>
    </tr>
    <tr>
        <td style="text-align: center;">
            <span style="font-weight: 500;" > {Cost}</span>
        </td>
        <td>
            <span>{Cancellation}</span>
        </td>
    </tr>
    <tr>
        <td style="text-align: center;">
            <span class="{Status}" style="font-weight: 500;" >
{Status}</span>
        </td>
    </tr>

```

```

        <td>
            <span style="font-weight: 500;color:
#0a76ff;">${Delivery}</span>
        </td>
    </tr>
</tbody>
</table>
</script>
<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Excel Export			
	Category	Product ID	Status
▶	Suits/Slim	EJ-SU-01	Available
▶	Suits/Classic	EJ-SU-02	Available
▶	Suits/Formal	EJ-SU-03	Available
▶	Phants/Slim	EJ-PH-01	Available
▶	Phants/Classic	EJ-PH-02	Available
▶	Shirts/Slim	EJ-SH-01	Available
▶	Shirts/Formal	EJ-SH-02	Available

Exporting with caption template

The Excel export feature enables exporting of Grid with a caption template to an Excel document.

In the following sample, the customized caption text is exported to Excel using [captionText](#) property in the [exportGroupCaption](#) event.

INDEX.TS

```

import { Grid, Group, Toolbar, ExcelExport, ExportGroupCaptionEventArgs }
  from '@syncfusion/ej2-grids';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { employeeData } from '../datasource.ts';
Grid.Inject(Group, Toolbar, ExcelExport);
let grid: Grid = new Grid({
  dataSource: employeeData,
  allowGrouping: true,
  groupSettings: { captionTemplate: '#captiontemplate', columns:
    ['EmployeeID'] },
  allowExcelExport: true,
  toolbar: ['ExcelExport'],
  columns: [
    { field: 'EmployeeID', headerText: 'Employee ID', width: 120 },
    { field: 'FirstName', headerText: 'Name', width: 120 },
    { field: 'City', headerText: 'City' },
    { field: 'Title', headerText: 'Title', width: 170 }
  ],
  toolbarClick: toolbarClick,
  exportGroupCaption: exportGroupCaption,
  height: 273
});
grid.appendTo('#Grid');
function toolbarClick(args: ClickEventArgs) {
  if (args.item.id === 'Grid_excelexport') {
    grid.excelExport();
  }
}

function exportGroupCaption(args: ExportGroupCaptionEventArgs) {
  args.captionText = args.data['field'] + ' - ' + args.data['key'];
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```



```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">


<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script id="captiontemplate" type="text/x-template">
            ${field} - ${key}
        </script>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Employ...  

 Excel Export

Name	City	Title
▼ EmployeeID - 1		
Nancy	Seattle	Sales Representative
▼ EmployeeID - 2		
Andrew	Tacoma	Vice President, Sales
▼ EmployeeID - 3		
Janet	Kirkland	Sales Representative
▼ EmployeeID - 4		

Exporting grid in server in EJ2 JavaScript Grid control

The Grid have an option to export the data to Excel in server side using Grid server export library.

Server dependencies

The Server side export functionality is shipped in the Syncfusion.EJ2.GridExport package, which is available in Essential Studio and [nuget.org](https://www.nuget.org). The following list of dependencies is required for Grid server side Excel exporting action.

- Syncfusion.EJ2
- Syncfusion.EJ2.GridExport
- Syncfusion.Compression.Base
- Syncfusion.XlsIO.Base

Server configuration

The following code snippets shows server configuration using ASP.NET MVC Controller Action.

To Export the Grid in server side, You need to call the [serverExcelExport](#) method for passing the Grid properties to server exporting action.

```
`ts
public ActionResult ExcelExport(string gridModel)
{
```



```

GridExcelExport exp = new GridExcelExport();
Grid gridProperty = ConvertGridObject(gridModel);
return exp.ExcelExport<OrdersDetails>(gridProperty, OrderRepository.GetAllRecords());
}

private Grid ConvertGridObject(string gridProperty)
{
    Grid GridModel = (Grid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty, typeof(Grid));
    GridColumnModel cols =
    (GridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
    typeof(GridColumnModel));
    GridModel.Columns = cols.columns;
    return GridModel;
}

public class GridColumnModel
{
    public List<GridColumn> columns { get; set; }
}

public ActionResult DataSource(DataManager dm)
{
    var DataSource = OrderRepository.GetAllRecords();
    DataResult result = new DataResult();
    result.result = DataSource.Skip(dm.Skip).Take(dm.Take).ToList();
    result.count = result.result.Count;
    return Json(result, JsonRequestBehavior.AllowGet);
}
`ts
import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Toolbar);
let data: DataManager = new DataManager({
    url: "Home/DataSource",
    adaptor: new UrlAdaptor
});

```

```

let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['ExcelExport'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
  ],
  height: 265
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_excelexport') {
    grid.serverExcelExport("Home/ExcelExport");
  }
}
`

```

Note: Refer to the GitHub sample for quick implementation and testing from [here](#).

CSV export in server side

You can export the Grid to CSV format by using the [serverCsvExport](#) method which will pass the Grid properties to server.

In the below demo, we have invoked the above method inside the [toolbarClick](#) event. In server side, we have deserialized the Grid properties and passed to the `CsvExport` method which will export the properties to CSV format.

```

`ts
public ActionResult CsvGridExport(string gridModel)
{
  GridExcelExport exp = new GridExcelExport();
  Grid gridProperty = ConvertGridObject(gridModel);
  return exp.CsvExport<OrdersDetails>(gridProperty, OrderRepository.GetAllRecords());
}

private Grid ConvertGridObject(string gridProperty)
{

```

```

Grid GridModel = (Grid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty, typeof(Grid));
GridColumnModel cols =
(GridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
typeof(GridColumnModel));
GridModel.Columns = cols.columns;
return GridModel;
}

public ActionResult DataSource(DataManager dm)
{
var DataSource = OrderRepository.GetAllRecords();
DataResult result = new DataResult();
result.result = DataSource.Skip(dm.Skip).Take(dm.Take).ToList();
result.count = result.result.Count;
return Json(result, JsonRequestBehavior.AllowGet);
}
、

`ts
import { Grid, Toolbar } from '@syncfusion/ej2-grids';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Toolbar);
let data: DataManager = new DataManager({
url: "Home/DataSource",
adaptor: new UrlAdaptor
});
let grid: Grid = new Grid({
dataSource: data,
toolbar: ['CsvExport'],
columns: [
{ field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', width: 100 },
{ field: 'CustomerID', headerText: 'Customer ID', width: 120 },
{ field: 'Freight', headerText: 'Freight', textAlign: 'Right', width: 120, format: 'C2' },
{ field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
],
height: 265

```

```
});
grid.appendTo('#Grid');
grid.toolbarClick = (args: Object) => {
  if (args['item'].id === 'Grid_csvexport') {
    grid.serverCsvExport("Home/CsvGridExport");
  }
}
`
```

Export grid as memory stream

The Grid offers an option to export the data as a memory stream instead of downloading it as a file in the browser. To obtain the memory stream of the exported grid, set the `AsMemoryStream` parameter to **true** in the [ExcelExport](#) and [CsvExport](#) methods.

The following code demonstrates how to get the memory stream of exported grid.

```
`ts
public object ExcelExport(string gridModel)
{
  GridExcelExport exp = new GridExcelExport();
  Grid gridProperty = ConvertGridObject(gridModel);
  // pass third parameter as true to get the Memory Stream of exported grid data
  return (MemoryStream)exp.ExcelExport<OrdersDetails>(gridProperty, OrderRepository.GetAllRecords(),
  true);
}

public object CsvExport(string gridModel)
{
  GridExcelExport exp = new GridExcelExport();
  Grid gridProperty = ConvertGridObject(gridModel);
  return (MemoryStream)exp.CsvExport<OrdersDetails>(gridProperty, OrderRepository.GetAllRecords(),
  true);
}
`
```

Merge grid's memory stream

The [Essential XlsIO](#) library is used to merge multiple memory streams into a single stream. To learn more about the merge option, please refer to this [documentation](#).

You can merge a memory stream, a file stream, and a local file with the Grid's memory stream in the following ways:

Merging with an existing memory stream

If you already have a memory stream, you can directly use it to merge with the Grid's memory stream.

In the following code, `ExcelEngine` and `AddCopy` method of `Worksheets` are used to merge the grid's memory stream with an existing memory stream.

```
`ts
using Syncfusion.XlsIO;

public MemoryStream ms1; // defines existing memory stream
public object ExcelExport(string gridModel)
{
    GridExcelExport exp = new GridExcelExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    // get the memory stream of exported grid data
    MemoryStream ms2 = (MemoryStream)exp.ExcelExport<OrdersDetails>(gridProperty,
    OrderRepository.GetAllRecords(), true);

    //New instance of ExcelEngine is created equivalent to launching Microsoft Excel with no workbooks
    open
    ExcelEngine excelEngine = new ExcelEngine();
    //Instantiate the Excel application object
    IApplication application = excelEngine.Excel;
    //Assigns default application version
    application.DefaultVersion = ExcelVersion.Xlsx;

    //open an workbook of existing memory stream and grid's memory stream through Open method of
    IWorkbooks
    IWorkbook sourceWorkbook = application.Workbooks.Open(ms1);
    IWorkbook destinationWorkbook = application.Workbooks.Open(ms2);
    //Copy all the worksheet from the Source workbook to the destination workbook
    for (int i = 0; i < sourceWorkbook.Worksheets.Count; i++)
    {
        destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[i]);
    }
    destinationWorkbook.ActiveSheetIndex = 1;
    //Saving the workbook as stream
    MemoryStream ms3 = new MemoryStream();
    destinationWorkbook.SaveAs(ms3);
    ms3.Position = 0;
```

```
//Dispose the instance of ExcelEngine
excelEngine.Dispose();
//Dispose the streams.
ms1.Dispose();
ms2.Dispose();
return ms3;
}
`ts
```

Merging with an existing file stream

If you already have a file stream, you can directly use it to merge with the Grid's memory stream. In the following code, the existing file stream is merged with the Grid's memory stream.

```
`ts
using Syncfusion.XlsIO;
public FileStream fs1; // defines existing file stream
public object ExcelExport(string gridModel)
{
    GridExcelExport exp = new GridExcelExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    MemoryStream ms1 = (MemoryStream)exp.ExcelExport<OrdersDetails>(gridProperty,
    OrderRepository.GetAllRecords(), true);
    ExcelEngine excelEngine = new ExcelEngine();
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Xlsx;
    //fs1 and ms1 represents the existing stream and grid's stream.
    IWorkbook sourceWorkbook = application.Workbooks.Open(fs1);
    IWorkbook destinationWorkbook = application.Workbooks.Open(ms1);
    for (int i = 0; i < sourceWorkbook.Worksheets.Count; i++)
    {
        destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[i]);
    }
    destinationWorkbook.ActiveSheetIndex = 1;
    //Saving the workbook as stream
    MemoryStream ms3 = new MemoryStream();
    destinationWorkbook.SaveAs(ms3);
}
```

```

ms3.Position = 0;
return ms3;
}
`

```

Merging with a local file

To merge a local file with the Grid's memory stream, you need to convert it into a file stream before merging. In the following code, the existing local file is merged with the Grid's memory stream.

```

`ts
using Syncfusion.XlsIO;

// get the file stream of local file
public FileStream fs1 = new FileStream("D:/ExcelDoc.xlsx", FileMode.Open, FileAccess.Read); //
ExcelDoc.xlsx is a local file which is located in local disk D.

public object ExcelExport(string gridModel)
{
    GridExcelExport exp = new GridExcelExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    MemoryStream ms1 = (MemoryStream)exp.ExcelExport<OrdersDetails>(gridProperty,
    OrderRepository.GetAllRecords(), true);
    ExcelEngine excelEngine = new ExcelEngine();
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Xlsx;
    //fs1 and ms1 represents the local file's stream and grid's stream.
    IWorkbook sourceWorkbook = application.Workbooks.Open(fs1);
    IWorkbook destinationWorkbook = application.Workbooks.Open(ms1);
    for (int i = 0; i < sourceWorkbook.Worksheets.Count; i++)
    {
        destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[i]);
    }
    destinationWorkbook.ActiveSheetIndex = 1;
    MemoryStream ms3 = new MemoryStream();
    destinationWorkbook.SaveAs(ms3);
    ms3.Position = 0;
    return ms3;
}

```

Downloading the merged memory stream

You can download the merged memory stream by converting it into a `FileStreamResult`. In the following code, the merged memory stream is downloaded to the browser.

```
`ts
using Syncfusion.XlsIO;

public ActionResult ExcelExport(string gridModel)
{
    ExcelEngine excelEngine = new ExcelEngine();
    IApplication application = excelEngine.Excel;
    application.DefaultVersion = ExcelVersion.Xlsx;
    //open an workbook of streams through Open method of IWorkbooks
    IWorkbook sourceWorkbook = application.Workbooks.Open(ms1);
    IWorkbook destinationWorkbook = application.Workbooks.Open(ms2);
    for (int i = 0; i < sourceWorkbook.Worksheets.Count; i++)
    {
        destinationWorkbook.Worksheets.AddCopy(sourceWorkbook.Worksheets[i]);
    }
    destinationWorkbook.ActiveSheetIndex = 1;
    MemoryStream ms3 = new MemoryStream();
    destinationWorkbook.SaveAs(ms3);
    ms3.Position = 0;
    // Save the MemoryStream into FileStreamResult
    FileStreamResult fileStreamResult = new FileStreamResult(ms3, "Application/vnd.openxmlformats-officedocument.spreadsheetml.sheet");
    fileStreamResult.FileNameDownload = "Export.xlsx";
    //Dispose the instance of ExcelEngine
    excelEngine.Dispose();
    //Dispose the streams.
    ms1.Dispose();
    ms2.Dispose();
    // return the file
    return fileStreamResult;
}
```


Rotate a header text to a certain degree in the exported grid on the server side

The DataGrid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported Excel file. To achieve this requirement, use the `ServerExcelHeaderQueryCellInfo` event of the Grid.

The `ServerExcelHeaderQueryCellInfo` will be triggered when creating a column header for the excel document to be exported in the server side. Customize the column header in this event.

In the following demo, using the `HeaderCellRotate` method of the `GridExcelExport` class in the `ServerExcelHeaderQueryCellInfo` event, you can rotate the header text of the column header in the excel exported document.

`ts

```
public ActionResult ExcelExport(string gridModel)
{
    GridExcelExport exp = new GridExcelExport();
    Grid gridProperty = ConvertGridObject(gridModel);
    gridProperty.ServerExcelHeaderQueryCellInfo = ExcelHeaderQueryCellInfo;
    IEnumerable data = Utils.DataTableToJson(dt);
    var result = exp.ExcelExport<dynamic>(gridProperty, data);
    return result;
}

private void ExcelHeaderQueryCellInfo(object excel)
{
    ServerExcelHeaderQueryCellInfoEventArgs name = (ServerExcelHeaderQueryCellInfoEventArgs)excel;
    headerValues.Add(name.Column.HeaderText);
    var longestString = headerValues.Where(s => s.Length == headerValues.Max(m => m.Length)).First();
    GridExcelExport exp = new GridExcelExport();
    var size = exp.ExcelTextSize(name.Style.Font.FontName, (float)name.Style.Font.Size, longestString);
    name.Cell.RowHeight = size.Width;
    exp.HeaderCellRotate(name, 45); // Give the rotate degree value by the user.
    name.Style.Borders.LineStyle = Syncfusion.XlsIO.ExcelLineStyle.None;
}
```

Limitations

- The export feature for detail templates is not supported in server-side exporting.
- Multiple grids exporting feature is not supported with server side exporting.

Global local in EJ2 JavaScript Grid control

Localization

The [Localization](#) library allows you to localize default text content of the Grid. The grid component has static text on some features (like group drop area text, pager information text, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

The following list of properties and its values are used in the grid.

Locale keywords | Text

EmptyRecord | No records to display

True | true

False | false

InvalidFilterMessage | Invalid Filter Data

GroupDropArea | Drag a column header here to group its column

UnGroup | Click here to ungroup

GroupDisable | Grouping is disabled for this column

FilterbarTitle | \s filter bar cell

EmptyDataSourceError | DataSource must not be empty at initial load since columns are generated from dataSource in AutoGenerate Column Grid

Add | Add

Edit | Edit

Cancel | Cancel

Update | Update

Delete | Delete

Print | Print

Pdfexport | PDF Export

Excelexport | Excel Export

Wordexport | Word Export

Csvexport | CSV Export

Search | Search

Columnchooser | Columns

Save | Save

Item | item

Items | items

EditOperationAlert | No records selected for edit operation

DeleteOperationAlert | No records selected for delete operation

SaveButton | Save

OKButton | OK

CancelButton | Cancel

EditFormTitle | Details of

AddFormTitle | Add New Record

BatchSaveConfirm | Are you sure you want to save changes?

BatchSaveLostChanges | Unsaved changes will be lost. Are you sure you want to continue?

ConfirmDelete | Are you sure you want to Delete Record?

CancelEdit | Are you sure you want to Cancel the changes?

ChooseColumns | Choose Column

SearchColumns | search columns

Matches | No Matches Found

FilterButton | Filter

ClearButton | Clear

StartsWith | Starts With

EndsWith | Ends With

Contains | Contains

Equal | Equal

NotEqual | Not Equal

LessThan | Less Than

LessThanOrEqual | Less Than Or Equal

GreaterThan | Greater Than

GreaterThanOrEqual | Greater Than Or Equal

ChooseDate | Choose a Date

EnterValue | Enter the value

Copy | Copy

Group | Group by this column

Ungroup | Ungroup by this column

autoFitAll | AutoFit all columns

autoFit | AutoFit this column

Export | Export

FirstPage | First Page

LastPage | Last Page

PreviousPage | Previous Page

NextPage | Next Page

SortAscending | Sort Ascending

SortDescending | Sort Descending

EditRecord | Edit Record

DeleteRecord | Delete Record

FilterMenu | Filter

SelectAll | Select All

Blanks | Blanks

FilterTrue | True

FilterFalse | False

NoResult | No Matches Found

ClearFilter | Clear Filter

NumberFilter | Number Filters

TextFilter | Text Filters

DateFilter | Date Filters

MatchCase | Match Case

Between | Between

CustomFilter | Custom Filter

CustomFilterPlaceholder | Enter the value

CustomFilterDatePlaceholder | Choose a date

AND | AND

OR | OR

ShowRowsWhere | Show rows where:

currentPageInfo | {0} of {1} pages

totalItemsInfo | ({0} items)

totalItemInfo | ({0} item)

firstPageTooltip | Go to first page

lastPageTooltip | Go to last page

nextPageTooltip | Go to next page

previousPageTooltip | Go to previous page

nextPagerTooltip | Go to next pager items

previousPagerTooltip | Go to previous pager items

pagerDropDown | Items per page

pagerAllDropDown | Items

All | All

Loading translations

To load translation object in an application, use [load](#) function of the [L10n](#) class.

The following example demonstrates the Grid in **Deutsch** culture.

INDEX.TS

```
import { L10n } from '@syncfusion/ej2-base';
import { Grid, Group, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group, Page);
L10n.load({
    'de-DE': {
        'grid': {
            'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
            'GroupDropArea': 'Ziehen Sie einen Spaltenkopf hier, um die
Gruppe ihre Spalte',
            'UnGroup': 'Klicken Sie hier, um die Gruppierung aufheben',
            'EmptyDataSourceError': 'DataSource darf bei der Erstaustausung
nicht leer sein, da Spalten aus der dataSource im AutoGenerate
Spaltenraster',
            'Item': 'Artikel',
            'Items': 'Artikel'
        },
        'pager': {
            'currentPageInfo': '{0} von {1} Seiten',
            'totalItemsInfo': '({0} Beiträge)',
            'firstPageTooltip': 'Zur ersten Seite',
            'lastPageTooltip': 'Zur letzten Seite',
            'nextPageTooltip': 'Zur nächsten Seite',
            'previousPageTooltip': 'Zurück zur letzten Seit',
            'nextPagerTooltip': 'Gehen Sie zu den nächsten Pager-Elementen',
            'previousPagerTooltip': 'Gehen Sie zu vorherigen Pager-
Elementen'
        }
    }
});
let grid: Grid = new Grid({
    dataSource: data,
    locale: 'de-DE',
    allowGrouping: true,
    allowPaging: true,
    pageSettings: { pageSize: 6 },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 210
});
```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Internationalization

The [Internationalization](#) library is used to globalize number, date, and time values in grid component using format strings in the [columns.format](#).

INDEX.TS

```

import { loadCldr, L10n, setCulture, setCurrencyCode } from
'@syncfusion/ej2-base';
import * as currencies from './currencies.json';
import * as cagregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
import * as numberingSystems from './numberingSystems.json';
import { Grid, Group, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group, Page);
loadCldr(currencies, cagregorian, numbers, timeZoneNames, numberingSystems);
setCulture('de');
setCurrencyCode('EUR');
L10n.load({
    'de-DE': {
        'grid': {
            'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
            'GroupDropArea': 'Ziehen Sie einen Spaltenkopf hier, um die
Gruppe ihre Spalte',
            'UnGroup': 'Klicken Sie hier, um die Gruppierung aufheben',
            'EmptyDataSourceError': 'DataSource darf bei der Erstaustauslastung
nicht leer sein, da Spalten aus der dataSource im AutoGenerate
Spaltenraster',
            'Item': 'Artikel',
            'Items': 'Artikel'
        },
        'pager': {
            'currentPageInfo': '{0} von {1} Seiten',
            'totalItemsInfo': '({0} Beiträge)',
            'firstPageTooltip': 'Zur ersten Seite',
            'lastPageTooltip': 'Zur letzten Seite',
            'nextPageTooltip': 'Zur nächsten Seite',

```

```

        'previousPageTooltip': 'Zurück zur letzten Seit',
        'nextPagerTooltip': 'Gehen Sie zu den nächsten Pager-Elementen',
        'previousPagerTooltip': 'Gehen Sie zu vorherigen Pager-
Elementen'
    }
}
});
let grid: Grid = new Grid({
    dataSource: data,
    locale: 'de-DE',
    allowGrouping: true,
    allowPaging: true,
    pageSettings: { pageSize: 6 },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        {
            field: 'Freight', headerText: 'Freight', width: 150, format: {
                format: 'C2', useGrouping: false,
                minimumSignificantDigits: 1, maximumSignificantDigits: 3,
currency: 'EUR'
            }, textAlign: 'Right'
        },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    height: 220
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* In the above sample, **Freight** column is formatted by **NumberFormatOptions**.

* By default, **locale** value is **en-US**. If you want to change the **en-US** culture to a different culture, you have to change the **locale** accordingly.

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Grid component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Grid, set the [enableRtl](#) to true.

INDEX.TS

```
import { L10n } from '@syncfusion/ej2-base';
import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
L10n.load({
  'ar-AE': {
    'grid': {
      'EmptyRecord': 'لا سجلات لعرضها',
      'EmptyDataSourceError': 'يجب أن يكون مصدر البيانات فارغة في  
التحميل الأولي منذ يتم إنشاء الأعمدة من مصدر البيانات في أوتوجينيراتد عمود  
الشبكة',
    },
    'pager': {
      'currentPageInfo': '{0} صفحة 1 {من}',
      'totalItemsInfo': '({0} العناصر)',
      'firstPageTooltip': 'انتقل إلى الصفحة الأولى',
      'lastPageTooltip': 'انتقل إلى الصفحة الأخيرة',
      'nextPageTooltip': 'انتقل إلى الصفحة التالية',
      'previousPageTooltip': 'انتقل إلى الصفحة السابقة',
      'nextPagerTooltip': 'انتقل إلى عناصر بيكر التالية',
      'previousPagerTooltip': 'للذهاب إلى عناصر بيكر السابقة'
    }
  }
});
let grid: Grid = new Grid({
  dataSource: data,
  enableRtl: true,
  locale: 'ar-AE',
  allowPaging: true,
  pageSettings: { pageSize: 7 },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Internationalization](#)
- [Localization](#)

Accessibility in EJ2 JavaScript Grid control

The Grid component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

WAI-ARIA attributes

The Grid component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Grid component:

Attributes	Purpose
---	---
<code>role=grid</code>	To represent the element containing the grid component.
<code>role=row</code>	To represent the element containing the cells of the row in the grid.
<code>role=rowgroup</code>	To represent the group of rows in the grid.
<code>role=columnheader</code>	To represent the cell in a row contains header information for a column in the grid.
<code>role=gridcell</code>	To represent a cell in the grid component.
<code>role=button</code>	To represent the element that acts as a button in the grid.
<code>role=search</code>	To represent the element that acts as a search region in the grid.
<code>role=presentation</code>	To represent the element to be not available for accessibility concerns.
<code>role=navigation</code>	To represent the element containing pager elements to navigate from one page to another.
<code>aria-colindex</code>	Defines the column index of the column with respect to the total number of columns within the grid.
<code>aria-rowindex</code>	Defines row index of the row with respect to the total number of rows within the grid.
<code>aria-rowspan</code>	Defines the number of rows spanned by a cell within the grid.
<code>aria-colspan</code>	Defines the number of columns spanned by a cell within the grid.
<code>aria-rowcount</code>	Defines the total number of rows in the grid.
<code>aria-colcount</code>	Defines the total number of columns in the grid.

- | **aria-selected** | Indicates the current "selected" state of the rows and cells in the grid. |
- | **aria-expanded** | Indicate if the expand icon in the hierarchy grid or grouped grid or detail grid is expanded or collapsed |
- | **aria-sort** | Indicates whether the data in the grid are sorted in ascending or descending order. |
- | **aria-busy** | Indicates an element is being modified and that assistive technologies may want to wait until the changes are complete before informing the user about the update. |
- | **aria-owns** | Identifies an element in order to define a visual, functional, or contextual relationship between a parent and its child elements. |
- | **aria-hidden** | Hides the element from accessibility concerns. |
- | **aria-labelledby** | Provides an accessible name for the checkbox labels in excel filter, checkbox filter and column chooser dialog. |
- | **aria-describedby** | Provides an description about the features enabled in the header when the grid header cell is focused. |

Keyboard interaction

The Grid component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Grid component.

Pager

Windows | MAC | To do this

Home | Fn + Left Arrow | Moves the focus to the first cell of the focused row.

End | Fn + Right Arrow | Moves the focus to the last cell of the focused row.

Ctrl + Home | Command + Fn + Left Arrow | Moves the focus to the first Cell of the first row in the grid.

Ctrl + End | Command + Fn + Right Arrow | Moves the focus to the last Cell of the last row in the grid.

Up Arrow | Up Arrow | Moves the cell focus upward from the focused cell.

Down Arrow | Down Arrow | Moves the cell focus downward from the focused cell.

Right Arrow | Right Arrow | Moves the cell focus right side from the focused cell.

Left Arrow | Left Arrow | Moves the cell focus left side from the focused cell.

Alt + J | Alt + J | Moves the focus to the entire grid.

Alt + W | Alt + W | Move the focus to the grid content element.

Selection

Windows | MAC | To do this

Ctrl + Up Arrow | Command + Up Arrow | Collapses all the visible groups.

Ctrl + Down Arrow | Command + Down Arrow | Expands all the visible groups.

Ctrl + Space | Ctrl + Space | Performs grouping when focused on a header element.

Enter | Enter | If the current cell is an expand/collapse cell then expands/collapses the current group/detailrow/childgrid.

Print

Windows | MAC | To do this

Ctrl + C | Command + C | Copies selected rows or cells data into the clipboard.

Ctrl + Shift + H | Ctrl + Shift + H | Copies selected rows or cells data with header into clipboard

Editing

Windows | MAC | To do this

Alt + Down arrow | Alt + Down arrow | Opens the filter menu(excel, menu and checkbox filter) when its header element is in focused state.

Column Menu

Windows | MAC | To do this

Ctrl + left arrow or right arrow | Command + left arrow or right arrow | Reorders the focused header column to the left or right side.

Sorting

Windows | MAC | To do this

Enter | Enter | Performs sorting(ascending/descending) on a column when its header element is in focused state.

Ctrl + Enter | Command + Enter | Performs multi-sorting on a column when its header element is in focused state.

Shift + Enter | Shift + Enter | Clears sorting for the focused header column.

* The Command and Control keys on Mac devices can be interchanged. When this switch occurs, use the Command key in place of the Control key and the Control key in place of the Command key for the above listed key interactions with Mac devices.

* For example, after switching the keys to group the columns when the header element is focused use Command + Space and for expanding the visible groups use Ctrl + Down Arrow.

Ensuring accessibility

The Grid component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Grid component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Grid component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

Clipboard in EJ2 JavaScript Grid control

The clipboard provides an option to copy selected rows or cells data into the clipboard.

The following list of keyboard shortcuts is supported in the Grid to copy selected rows or cells data into the clipboard.

Interaction keys | Description

Ctrl + C | Copy selected rows or cells data into clipboard.

Ctrl + Shift + H | Copy selected rows or cells data with header into clipboard.

INDEX.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let grid: Grid = new Grid({
  dataSource: data,
  allowSelection: true,
  selectionSettings: { type: 'Multiple' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 272
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Copy to clipboard by external buttons

To copy selected rows or cells data into the clipboard with help of external buttons, you need to invoke the [copy](#) method.

INDEX.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from './datasource.ts';

```



```

let grid: Grid = new Grid({
  dataSource: data,
  allowSelection: true,
  selectionSettings: { type: 'Multiple' },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  height: 280
});
grid.appendTo('#Grid');
let copyBtn: Button = new Button();
copyBtn.appendTo('#copy');
document.getElementById('copy').addEventListener('click', () => {
  grid.copy();
});
let copyHeaderBtn: Button = new Button();
copyHeaderBtn.appendTo('#copyHeader');
document.getElementById('copyHeader').addEventListener('click', () => {
  grid.copy(true);
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="copy">Copy</button>
        <button id="copyHeader">Copy With Header</button>
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

AutoFill

AutoFill Feature allows you to copy the data of selected cells and paste it to another cells by just dragging the autofill icon of the selected cells up to required cells. This feature is enabled by defining `enableAutoFill` property as true.

INDEX.TS

```

import { Grid, Selection, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Selection, Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    enableAutoFill: true,
    enableHover: false,
    allowSelection: true,
    toolbar: ['Add', 'Update', 'Cancel'],
    selectionSettings: { type: 'Multiple', mode: 'Cell', cellSelectionMode:
'Box' },
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Batch' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
isPrimaryKey: true, visible: false, width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'ShipCity', width: 150 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ]
});

```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* If `enableAutoFill` is set to true, then the autofill icon will be displayed on cell selection to copy cells.

* It requires the selection `mode` to be `Cell`, `cellSelectionMode` to be `Box` and also Batch Editing should be enabled.

Limitations of AutoFill

- Since the string values are not parsed to number and date type, so when the selected string type cells are dragged to number type cells then it will display as **NaN**. For date type cells, when the selected string type cells are dragged to date type cells then it will display as an **empty cell**.
- Linear series and the sequential data generations are not supported in this autofill feature.

Paste

You can able to copy the content of a cell or a group of cells by selecting the cells and pressing Ctrl + C shortcut key and paste it to another set of cells by selecting the cells and pressing Ctrl + V shortcut key.

INDEX.TS

```

import { Grid, Selection, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Selection, Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    enableHover: false,
    allowSelection: true,
    toolbar: ['Add', 'Update', 'Cancel'],
    selectionSettings: { type: 'Multiple', mode: 'Cell', cellSelectionMode: 'Box' },
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Batch' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right', isPrimaryKey: true, visible: false, width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'ShipCity', headerText: 'ShipCity', width: 150 },
    ]
});

```

```

        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ];
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>

```

```

    }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To perform paste functionality, it requires the selection **mode** to be **Cell**, **cellSelectionMode** to be **Box** and also Batch Editing should be enabled.

Limitations of Paste Functionality

- Since the string values are not parsed to number and date type, so when the copied string type cells are pasted to number type cells then it will display as **NaN**. For date type cells, when the copied string format cells are pasted to date type cells then it will display as an **empty cell**.

Context menu in EJ2 JavaScript Grid control

The Grid has options to show the context menu when right clicked on it. To enable this feature, you need to define either default or custom item in the [contextMenuItems](#).

To use the context menu, inject the **ContextMenu** module in the grid.

The default items are in the following table.

Items | Description

AutoFit | Auto fit the current column.

AutoFitAll | Auto fit all columns.

Edit | Edit the current record.

Delete | Delete the current record.

Save | Save the edited record.

Cancel | Cancel the edited state.

Copy | Copy the selected records.

PdfExport | Export the grid data as Pdf document.

ExcelExport | Export the grid data as Excel document.

CsvExport | Export the grid data as CSV document.

Group | Group the current column.

Ungroup | Ungroup the current column.

SortAscending | Sort the current column in ascending order.

SortDescending | Sort the current column in descending order.

FirstPage | Go to the first page.

PrevPage | Go to the previous page.

LastPage | Go to the last page.

NextPage | Go to the next page.

INDEX.TS

```
import { Grid, Resize, Sort, Group, ContextMenu, Edit, Page, PdfExport,
ExcelExport } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Resize, Sort, Group, Edit, ContextMenu, Page, PdfExport,
ExcelExport);
let grid: Grid = new Grid({
  dataSource: data,
  allowSorting: true,
  allowPaging: true,
  editSettings: {allowEditing: true, allowDeleting: true},
  allowPdfExport: true,
  allowExcelExport: true,
  contextMenuItems: ['AutoFit', 'AutoFitAll', 'SortAscending',
'SortDescending',
'Copy', 'Edit', 'Delete', 'Save', 'Cancel',
'PdfExport', 'ExcelExport', 'CsvExport', 'FirstPage',
'PrevPage',
'LastPage', 'NextPage'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', width: 200, textAlign:
'Right'},
    { field: 'Freight', width: 150, format: 'C2', textAlign: 'Right',
editType: 'numericedit' },
    { field: 'ShipName', headerText: 'Ship Name', width: 300 },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 200 },
    { field: 'ShipCity', headerText: 'Ship City', width: 200 }
  ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

```



```

        <tr>
            <td class="CardHeader">Title
            </td>
            <td>${Title}
            </td>
        </tr>
        <tr>
            <td class="CardHeader">Country
            </td>
            <td>${Country}
            </td>
        </tr>
    </tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom context menu items

The custom context menu items can be added by defining the [contextMenuItems](#) as a collection of [contextMenuItemModel](#). Actions for this customized items can be defined in the [contextMenuClick](#) event.

INDEX.TS

```

import { Grid, ContextMenu, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(ContextMenu, Page);
let grid: Grid = new Grid({
    dataSource: data,
    contextMenuItems: [{text: 'Copy with headers', target: '.e-content' id:
'copywithheader'}],
    allowSelection: true,
    allowPaging: true,
    contextMenuClick: function(args: MenuEventArgs){
        if(args.item.id === 'copywithheader'){
            grid.copy(true);
        }
    },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Left',
width: 125, isPrimaryKey: true },

```

```

        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 125 },
        { field: 'ShipName', headerText: 'Ship Name', width: 120 },
        { field: 'ShipCity', headerText: 'Ship City', width: 170 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150,
textAlign: 'Right' }
    ]
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script id="rowtemplate" type="text/x-template">
        <tr>
            <td class="photo">

```

```

        
    </td>
    <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
            <colgroup>
                <col width="50%">
                <col width="50%">
            </colgroup>
            <tbody>
                <tr>
                    <td class="CardHeader">First Name </td>
                    <td>${FirstName} </td>
                </tr>
                <tr>
                    <td class="CardHeader">Last Name</td>
                    <td>${LastName} </td>
                </tr>
                <tr>
                    <td class="CardHeader">Title
                    </td>
                    <td>${Title}
                    </td>
                </tr>
                <tr>
                    <td class="CardHeader">Country
                    </td>
                    <td>${Country}
                    </td>
                </tr>
            </tbody>
        </table>
    </td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show context menu on left click

By default, the context menu items will be shown in the Grid using the right mouse click action. Show the context menu items during the left mouse click action using the [created](#) and context menu's `beforeOpen` events of the Grid.

Using the `onclick` eventlistener of Grid , you can get the clicked position values and send them to the `open` method of the context menu in the `onclick` event of the Grid. Also, we have prevented the default right click action to open the context menu items using the [created](#) event of the Grid.

This is demonstrated in the following sample.

INDEX.TS

```
import { Grid, Edit, Page, ContextMenu } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Page, ContextMenu);
let values: any;
let grid: Grid = new Grid({
  dataSource: data,
  editSettings: { allowAdding: true, allowDeleting: true, allowEditing: true
},
  allowPaging: true,
  contextMenuItems: ['Copy', 'Edit', 'Delete'],
  columns: [
    {
      field: 'OrderID',
      headerText: 'Order ID',
      width: 120,
      textAlign: 'Right',
      isPrimaryKey: true,
    },
    { field: 'CustomerName', headerText: 'Customer Name' },
    {
      field: 'Freight',
      format: 'C2',
      textAlign: 'Right',
      editType: 'numericedit',
    },
    { field: 'ShipName', headerText: 'Ship Name', width: 200 },
    {
      field: 'ShipCountry',
      headerText: 'Ship Country',
      width: 150,
      editType: 'dropdownedit',
    },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
  ],
  created: () => {
    grid.contextMenuModule.contextMenu.beforeOpen = (args) => {
      if (args.event && args.event.which === 3) args.cancel = true;
      args.event = values;
      grid.contextMenuModule.contextMenuBeforeOpen(args);
    };
  },
});
grid.appendTo('#Grid');
document.getElementById('Grid').onclick = (event) => {
  values = event;
  grid.contextMenuModule.contextMenu.open(
    values.pageY + pageYOffset,
    values.pageX + pageXOffset
  );
};
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>

```

```

        <tr>
            <td class="CardHeader">Last Name</td>
            <td>${LastName} </td>
        </tr>
        <tr>
            <td class="CardHeader">Title
            </td>
            <td>${Title}
            </td>
        </tr>
        <tr>
            <td class="CardHeader">Country
            </td>
            <td>${Country}
            </td>
        </tr>
    </tbody>
</table>
</td>
</tr>
</script>

<div id="container">
    <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can hide or show an item in context menu for specific area inside of grid by defining the [target](#) property.

Enable or disable context menu items

It is possible to enable or disable the default and custom context menu items in the Grid component. This is achieved by using the [enableItems](#) method of the ContextMenu. To enable or disable menu items, set the `enable` parameter in the `enableItems` method to true, and vice versa.

In the following sample, the Copy item is enabled or disabled based on some condition (as per the needs of the application) in the [rowSelected](#) event of the Grid.

INDEX.TS

```

import { Grid, Edit, Page, ContextMenu } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Page, ContextMenu);
let grid: Grid = new Grid({
    dataSource: data,
    editSettings: { allowAdding: true, allowDeleting: true, allowEditing:
true },
    allowPaging: true,
    contextMenuItems: ['Copy', 'Edit', 'Delete'],

```

```

        rowSelected: rowSelected,
        columns: [
            { field: 'OrderID', headerText: 'Order ID', width: 120, textAlign:
'Right', isPrimaryKey: true },
            { field: 'CustomerName', headerText: 'Customer Name' },
            { field: 'Freight', format: 'C2', textAlign: 'Right', editType:
'numericedit' },
            { field: 'ShipName', headerText: 'Ship Name', width: 200 },
            { field: 'ShipCountry', headerText: 'Ship Country', width: 150,
editType: 'dropdownedit' },
            { field: 'ShipCity', headerText: 'Ship City', width: 150 }
        ]
    });
    grid.appendTo('#Grid');
    function rowSelected(args) {
        var contextMenuObj = grid.contextMenuModule.contextMenu;
        if (args.data.OrderID % 2 === 0) {
            contextMenuObj.enableItems(['Copy'], false);
        } else {
            contextMenuObj.enableItems(['Copy'], true);
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="rowtemplate" type="text/x-template">
    <tr>
      <td class="photo">
        
      </td>
      <td class="details">
        <table class="CardTable" cellpadding="3" cellspacing="2">
          <colgroup>
            <col width="50%">
            <col width="50%">
          </colgroup>
          <tbody>
            <tr>
              <td class="CardHeader">First Name </td>
              <td>{FirstName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Last Name</td>
              <td>{LastName} </td>
            </tr>
            <tr>
              <td class="CardHeader">Title
              </td>
              <td>{Title}
              </td>
            </tr>
            <tr>
              <td class="CardHeader">Country
              </td>
              <td>{Country}
              </td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </script>

  <div id="container">
    <div id="Grid"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>

```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Loading animation in EJ2 JavaScript Grid control

The grid has an option to show a loading indicator in-between the time of fetching the data and binding it to the grid during initial rendering or refreshing or after performing any grid action like sorting, filtering, grouping, and more. The grid supports two indicator types, which is achieved by setting the `loadingIndicator.indicatorType` property to `Spinner` or `Shimmer`. The default value of the indicator type is `"Spinner"`.

In the following sample, the Shimmer indicator is displayed while the grid is loading and refreshing when using the remote data.

INDEX.TS

```
import { Grid, Page, Sort, Filter } from '@syncfusion/ej2-grids';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
Grid.Inject(Page, Sort, Filter);
let data: DataManager = new DataManager({
    url: 'https://services.syncfusion.com/js/production/api/Orders',
    adaptor: new ODataAdaptor
});
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    allowSorting: true,
    allowFiltering: true,
    pageSettings: { pageCount: 3 },
    loadingIndicator: { indicatorType: 'Shimmer' },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd' }
    ]
});
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Style And Appearance

Style and appearance in EJ2 JavaScript Grid control

The Grid control offers various ways to customize its appearance using both default CSS and custom themes. Let's go over some common approaches:

Default CSS overrides:

You can use custom CSS to override the default styles of the Grid control. This allows you to change colors, fonts, paddings, and more. You can inspect the generated HTML of the Grid using browser developer tools to identify the relevant CSS classes and styles.

Here's a basic example of how you can override the header background color of the Grid:

```
`css
```

```
/ In your control's CSS file /
```

```
.e-grid .e-headercell {
```

```
background-color: #333; / Override the header background color /
```

```
color: #fff;
```

```
}
```

```
`
```

ORDER ID ▾	CUSTOMER ID ▾	FREIGHT ▾	ORDER DATE ▾
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996
10252	SUPRD	\$51.30	7/9/1996
10253	HANAR	\$58.17	7/10/1996

Using theme studio:

Syncfusion's Theme Studio tool allows you to create custom themes for all their controls, including the Grid. This is a more advanced approach that lets you define a comprehensive set of styles to achieve a consistent look and feel throughout your application.

1. Visit the [Syncfusion Theme Studio](#).
2. Select the Grid control from the left panel.
3. Customize various aspects of the control's appearance, such as colors, typography, and spacing.
4. Once done, you can download the generated CSS file and include it in your EJ2 JavaScript project.

Customizing the grid root element

To customize the appearance of the root element of the Syncfusion EJ2 JavaScript Grid control, you can use CSS. Here's an example of how to modify the font family and row colors using CSS:

```
`css
.e-grid {
font-family: cursive;
}
`
```

ORDER ID ▾	CUSTOMER ID ▾	FREIGHT ▾	ORDER DATE ▾
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996
10252	SUPRD	\$51.30	7/9/1996
10253	HANAR	\$58.17	7/10/1996

The above code snippet, the **.e-grid** class targets the root element of the Syncfusion EJ2 JavaScript Grid control, and the **font-family** property is set to cursive to change the font family of the grid content.

In the following sample, the font family of grid content is changed to **cursive**, and the background color of rows, selected rows, alternate rows, and row hovering color is modified using the below CSS styles.

INDEX.TS

```
import { Grid, Page, Selection } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Selection);
let grid: Grid = new Grid({
  dataSource: data,
  selectionSettings: { type: 'Multiple' },
  allowPaging: true,
  pageSettings: { pageSize: 8 },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', type: 'number',
      textAlign: 'Right', isPrimaryKey: true, width: 100 },
    { field: 'CustomerID', headerText: 'Customer ID', type: 'string',
      width: 120 },
    { field: 'Freight', headerText: 'Freight', type: 'number', format:
      'C2', textAlign: 'Right', width: 100 },
    { field: 'ShipName', headerText: 'Ship Name', type: 'string', width:
      180 },
  ],
  height: 273
});
```

```
grid.appendTo('#Grid');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en">
<head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <style>
    .e-grid {
      font-family: cursive;
    }
    .e-grid .e-row:hover .e-rowcell {
      background-color: rgb(204, 229, 255) !important;
    }
    .e-grid .e-rowcell.e-selectionbackground {
      background-color: rgb(230, 230, 250);
    }
    .e-grid .e-row.e-altrow {
      background-color: rgb(150, 212, 212);
    }
    .e-grid .e-row {
      background-color: rgb(180, 180, 180);
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

<div id="container" style="height:350px;">
  <div id="Grid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to change the font size of Grid elements](#)

Header in EJ2 JavaScript Grid control

You can customize the appearance of the header elements in the Syncfusion EJ2 JavaScript Grid control using CSS. Here are examples of how to customize the Grid header, header cell, and header cell div element.

Customizing the grid header

To customize the appearance of the Grid header root element, you can use the following CSS code:

```

`css
.e-grid .e-gridheader {
border: 2px solid green;
}
`

```

In this example, the **.e-gridheader** class targets the Grid header root element. You can modify the **border** property to change the style of the header border. This customization allows you to override the thin line between the header and content of the grid.

ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996

Customizing the grid header cell

To customize the appearance of the grid header cell elements, you can use the following CSS code:

```

`css
.e-grid .e-headercell {

```

```
color: #ffffff;
background-color: #1ea8bd;
}
```

In this example, the **.e-headercell** class targets the header cell elements. You can modify the **color** and **background-color** properties to change the text color and background of the header cells.

ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996

Customizing the grid header cell div element

To customize the appearance of the grid header cell div element, you can use the following CSS code:

```
`css
.e-grid .e-headercelldiv {
font-size: 15px;
font-weight: bold;
color: darkblue;
}
```

In this example, the **.e-headercelldiv** class targets the div element within the header cell. You can modify the **font-size**, **font-weight**, **color** properties to change the font size, font-weight and color of the header text content.

ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996

Paging in EJ2 JavaScript Grid control

You can customize the appearance of the paging elements in the Syncfusion EJ2 JavaScript Grid control using CSS. Here are examples of how to customize the pager root element, pager container element, pager navigation elements, pager page numeric link elements, and pager current page numeric element.

Customizing the grid pager root element

To customize the appearance of the grid pager root element, you can use the following CSS code:

```
`css
.e-grid .e-gridpager {
font-family: cursive;
background-color: #deecf9;
}
```

In this example, the **.e-gridpager** class targets the pager root element. You can modify the **font-family** to change the font family and **background-color** property to change the background color of the pager.

ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996
10252	SUPRD	\$51.30	7/9/1996
10253	HANAR	\$58.17	7/10/1996

« < 1 2 3 > »
1 of 3 pages (15 items)

Customizing the grid pager container element

To customize the appearance of the grid pager container element, you can use the following CSS code:

```
`css
.e-grid .e-pagercontainer {
border: 2px solid #00b5ff;
font-family: cursive;
}
```


In this example, the **.e-pagercontainer** class targets the pager container element. You can modify the **border** property and **font-family** property to change the border color and font family of the pager container.

ORDER ID	CUSTOMER NAME	ORDER DATE	FREIGHT	SHIPPED DATE
10248	Paul Henriot	7/4/1996	\$32.38	7/16/1996
10249	Karin Josephs	7/5/1996	\$11.61	7/10/1996
10250	Mario Pontes	7/8/1996	\$65.83	7/12/1996
10251	Mary Saveley	7/8/1996	\$41.34	7/15/1996
10252	Pascale Cartrain	7/9/1996	\$51.30	7/11/1996

<< < 1 2 3 4 5 ... > >>
1 of 166 pages (830 items)

Customizing the grid pager navigation elements

To customize the appearance of the grid pager navigation elements, you can use the following CSS code:

```
`css
.e-grid .e-gridpager .e-prevpagedisabled,
.e-grid .e-gridpager .e-prevpage,
.e-grid .e-gridpager .e-nextpage,
.e-grid .e-gridpager .e-nextpagedisabled,
.e-grid .e-gridpager .e-lastpagedisabled,
.e-grid .e-gridpager .e-lastpage,
.e-grid .e-gridpager .e-firstpage,
.e-grid .e-gridpager .e-firstpagedisabled {
background-color: #deecf9;
}
```

In this example, the classes **.e-prevpagedisabled**, **.e-prevpage**, **.e-nextpage**, **.e-nextpagedisabled**, **.e-lastpagedisabled**, **.e-lastpage**, **.e-firstpage**, and **.e-firstpagedisabled** target the various pager navigation elements. You can modify the **background-color** property to change the background color of these elements.

ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996
10252	SUPRD	\$51.30	7/9/1996
10253	HANAR	\$58.17	7/10/1996

<< < 1 2 3 > >>

1 of 3 pages (15 items)

Customizing the grid pager page numeric link elements

To customize the appearance of the grid pager current page numeric link elements, you can use the following CSS code:

```
`css
.e-grid .e-gridpager .e-numericitem {
background-color: #5290cb;
color: #ffffff;
cursor: pointer;
}
.e-grid .e-gridpager .e-numericitem:hover {
background-color: white;
color: #007bff;
}
`
```

In this example, the **.e-numericitem** class targets the page numeric link elements. You can modify the **background-color**, **color** properties to change the background color and text color of these elements.

ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996
10252	SUPRD	\$51.30	7/9/1996
10253	HANAR	\$58.17	7/10/1996

« < 1 2 3 > »
1 of 3 pages (15 items)

Customizing the grid pager current page numeric element

To customize the appearance of the grid pager current page numeric element, you can use the following CSS code:

```
`css
.e-grid .e-gridpager .e-currentitem {
background-color: #0078d7;
color: #fff;
}
```

In this example, the **.e-currentitem** class targets the current page numeric item. You can modify the **background-color** property to change the background color of this element and **color** property to change the text color.

ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996
10252	SUPRD	\$51.30	7/9/1996
10253	HANAR	\$58.17	7/10/1996

« < 1 2 3 > »
1 of 3 pages (15 items)

Sorting in EJ2 JavaScript Grid control

You can customize the appearance of the sorting icons and multi sorting icons in the Syncfusion EJ2 JavaScript Grid control using CSS. You can use the available Syncfusion [icons](#) based on your theme. Here's how to do it:

Customizing the grid sorting icon

To customize the sorting icon that appears in the Grid header when sorting is applied, you can use the following CSS code:

```
`css
.e-grid .e-icon-ascending::before {
content: '\e7a3'; / Icon code for ascending order /
}
.e-grid .e-icon-descending::before {
content: '\e7b6'; / Icon code for descending order /
}
`
```

In this example, the **.e-icon-ascending::before** class targets the sorting icon for ascending order, and the **.e-icon-descending::before** class targets the sorting icon for descending order.

ORDER ID	CUSTOMER ID	↓↑	FREIGHT	ORDER DATE
10259	CENTC		\$3.25	7/18/1996
10254	CHOPS		\$22.98	7/11/1996
10258	ERNSH		\$140.51	7/17/1996
10250	HANAR		\$65.83	7/8/1996
10253	HANAR		\$58.17	7/10/1996
10257	HILAA		\$81.91	7/16/1996

« < 1 2 3 > »
1 of 3 pages (15 items)

Customizing the grid multi sorting icon

To customize the multi sorting icon that appears in the grid header when multiple columns are sorted, you can use the following CSS code:

```
`css
.e-grid .e-sortnumber {
background-color: #deecf9;
font-family: cursive;
}
```

```
}
,
```

In this example, the **.e-sortnumber** class targets the background color and font family of the multi sorting icon. You can modify the **background-color** and **font-family** properties to customize the appearance of the multi sorting icon.

ORDER ID	CUSTOMER ID	1 ↑	FREIGHT ↓ 2	ORDER DATE
10259	CENTC		\$3.25	7/18/1996
10254	CHOPS		\$22.98	7/11/1996
10258	ERNSH		\$140.51	7/17/1996
10253	HANAR		\$58.17	7/10/1996
10250	HANAR		\$65.83	7/8/1996
10257	HILAA		\$81.91	7/16/1996

« < 1 2 3 > »
1 of 3 pages (15 items)

Filtering in EJ2 JavaScript Grid control

You can customize the appearance of filtering elements in the Syncfusion EJ2 JavaScript Grid control using CSS. Below are examples of how to customize various filtering elements, including filter bar cell elements, filter bar input elements, focus styles, clear icons, filter icons, filter dialog content, filter dialog footer, filter dialog input elements, filter dialog button elements, and Excel filter dialog number filters.

Customizing the filter bar cell element

To customize the appearance of the filter bar cell element in the grid header, you can use the following CSS code:

```
`css
.e-grid .e-filterbarcell {
background-color: #045fb4;
}
,
```

In this example, the **.e-filterbarcell** class targets the filter bar cell element in the grid header. You can modify the **background-color** property to change the color of the filter bar cell element.

ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996
10252	SUPRD	\$51.30	7/9/1996
10253	HANAR	\$58.17	7/10/1996

Customizing the filter bar input element

To customize the appearance of the filter bar input element in the grid header, you can use the following CSS code:

```
`css
.e-grid .e-filterbarcell .e-input-group input.e-input{
font-family: cursive;
}
`
```

In this example, the **.e-filterbarcell** class targets the filter bar cell element, and the **.e-input** class targets the input element within the cell. You can modify the **font-family** property to change the font of the filter bar input element.

ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
	HANAR ×		
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996
10252	SUPRD	\$51.30	7/9/1996
10253	HANAR	\$58.17	7/10/1996

Customizing the filter bar input focus

To customize the appearance of the filter bar input element's focus highlight, you can use the following CSS code:

```
`css
.e-grid .e-filterbarcell .e-input-group.e-input-focus{
background-color: #deecf9;
}
`
```

In this example, the **.e-filterbarcell** class targets the filter bar cell element, and the **.e-input-group.e-input-focus** class targets the focused input element. You can modify the `background-color` property to change the color of the focus highlight.


ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
	<input type="text"/>		
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996
10252	SUPRD	\$51.30	7/9/1996
10253	HANAR	\$58.17	7/10/1996

Customizing the filter bar input clear icon

To customize the appearance of the filter bar input element's clear icon, you can use the following CSS code:

```
`css
.e-grid .e-filterbarcell .e-input-group .e-clear-icon::before {
content: '\e72c';
}
`
```

In this example, the **.e-clear-icon** class targets the clear icon element within the input group. You can modify the `content` property to change the icon displayed.





ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
	VIN 		
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996
10252	SUPRD	\$51.30	7/9/1996
10253	HANAR	\$58.17	7/10/1996

Customizing the grid filtering icon

To customize the appearance of the grid's filtering icon in the grid header, you can use the following CSS code:

```
`css
.e-grid .e-icon-filter::before{
content: '\e81e';
}
```

In this example, the **.e-icon-filter** class targets the filtering icon element. You can modify the **content** property to change the icon displayed.

ORDER ID 	CUSTOMER ID 	FREIGHT 	ORDER DATE 
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996
10252	SUPRD	\$51.30	7/9/1996
10253	HANAR	\$58.17	7/10/1996

Customizing the filter dialog content

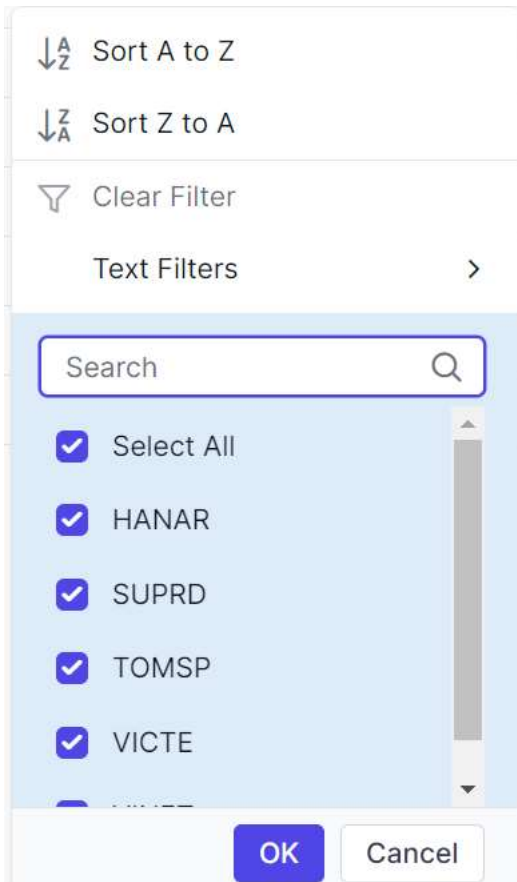
To customize the appearance of the filter dialog's content element, you can use the following CSS code:

```
`css
.e-grid .e-filter-popup .e-dlg-content {
```



```
background-color: #deecf9;
}
`
```

In this example, the **.e-filter-popup .e-dlg-content** classes target the content element within the filter dialog. You can modify the `background-color` property to change the color of the dialog's content.

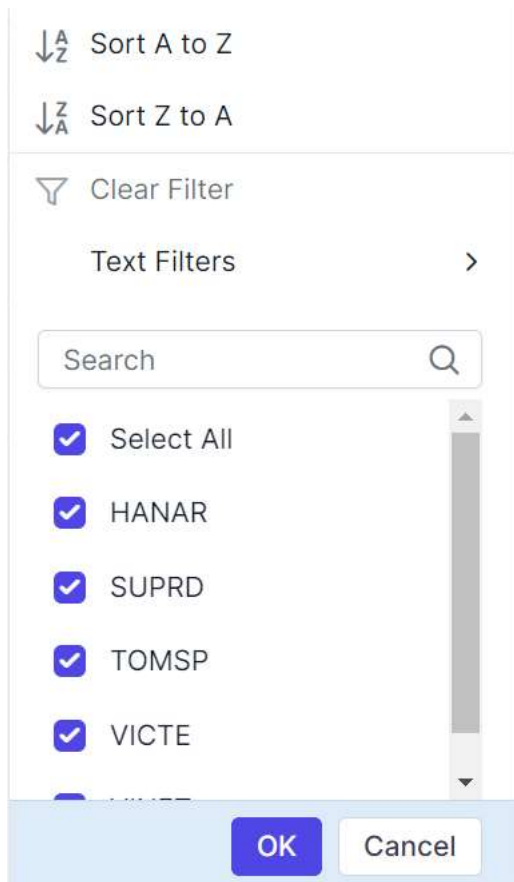


Customizing the filter dialog footer

To customize the appearance of the filter dialog's footer element, you can use the following CSS code:

```
`css
.e-grid .e-filter-popup .e-footer-content {
background-color: #deecf9;
}
`
```

In this example, the **.e-filter-popup .e-footer-content** classes target the footer element within the filter dialog. You can modify the `background-color` property to change the color of the dialog's footer.

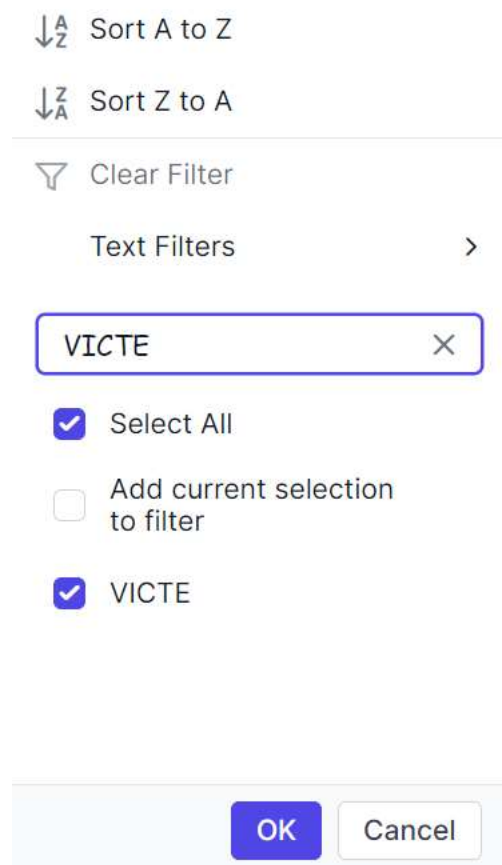


Customizing the filter dialog input element

To customize the appearance of the filter dialog's input elements, you can use the following CSS code:

```
`css
.e-grid .e-filter-popup .e-input-group input.e-input{
font-family: cursive;
}
```

In this example, the **.e-filter-popup** class targets the filter dialog, and the **.e-input** class targets the input elements within the dialog. You can modify the **font-family** property to change the font of the input elements.

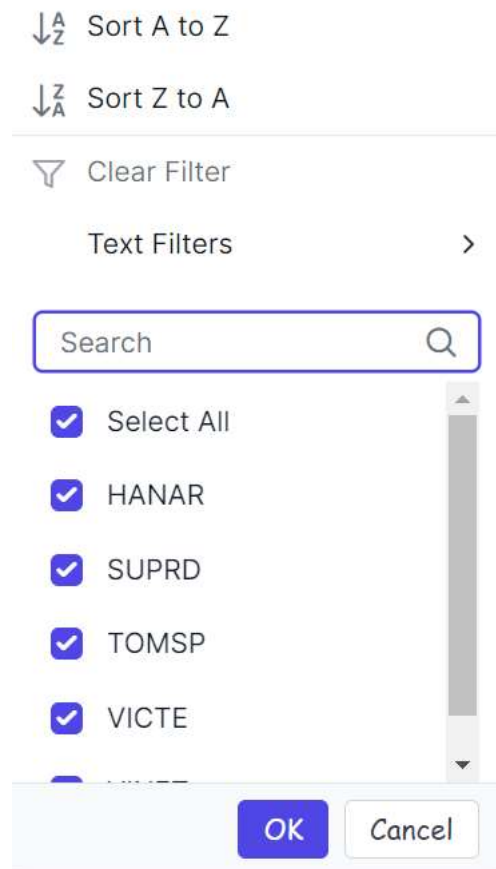


Customizing the filter dialog button element

To customize the appearance of the filter dialog's button elements, you can use the following CSS code:

```
`css
.e-grid .e-filter-popup .e-btn{
font-family: cursive;
}
`
```

In this example, the **.e-filter-popup** class targets the filter dialog, and the **.e-btn** class targets the button elements within the dialog. You can modify the `font-family` property to change the font of the button elements.

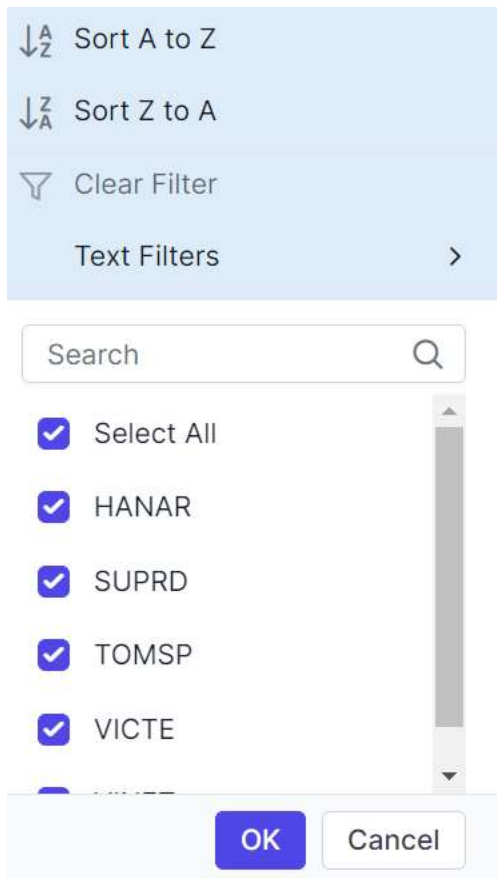


Customizing the excel filter dialog number filters element

To customize the appearance of the excel filter dialog's number filters, you can use the following CSS code:

```
`css
.e-grid .e-filter-popup .e-contextmenu-wrapper ul{
background-color: #deecf9;
}
`
```

In this example, the **.e-filter-popup .e-contextmenu-wrapper** ul classes target the number filter elements within the excel filter dialog. You can modify the `background-color` property to change the color of these elements.



Grouping in EJ2 JavaScript Grid control

You can customize the appearance of grouping elements in the Syncfusion EJ2 JavaScript Grid control using CSS. Here are examples of how to customize the group header, group expand/collapse icons, group caption row, and grouping indent cell.

Customizing the group header

To customize the appearance of the group header element, you can use the following CSS code:

```
`css
.e-grid .e-groupdroparea {
background-color: #132f49;
}
`
```

In this example, the **.e-groupdroparea** class targets the group header element. You can modify the **background-color** property to change the color of the group header.

Drag a column header here to group its column			
ORDER ID	CUSTOMER ID	SHIP CITY	SHIP NAME
10248	VINET	Reims	Vins et alcools Cheva...
10249	TOMSP	Münster	Toms Spezialitäten
10250	HANAR	Rio de Janeiro	Hanari Carnes
10251	VICTE	Lyon	Victuailles en stock

Customizing the group expand or collapse icons

To customize the appearance of the group expand/collapse icons in the grid, you can use the following CSS code:

```
`css
.e-grid .e-icon-gdownarrow::before{
content:'\e7c9'
}
.e-grid .e-icon-grightarrow::before{
content:'\e80f'
}
`
```

In this example, the **.e-icon-gdownarrow** and **.e-icon-grightarrow** classes target the expand and collapse icons, respectively. You can modify the `content` property to change the icon displayed. You can use the available Syncfusion icons based on your theme.

Customer ID ↑ ×		
ORDER ID	SHIP CITY	SHIP NAME
Customer ID: CENTC - 1 item		
Customer ID: CHOPS - 1 item		
Customer ID: ERNSH - 1 item		
Customer ID: HANAR - 2 items		
10250	Rio de Janeiro	Hanari Carnes
10253	Rio de Janeiro	Hanari Carnes
Customer ID: HILAA - 1 item		
10257	San Cristóbal	HILARION-Abastos

Customizing the group caption row

To customize the appearance of the group caption row and the icons indicating record expansion or collapse, you can use the following CSS code:

```
`css
.e-grid .e-groupcaption {
background-color: #deecf9;
}
.e-grid .e-recordplusexpend,
.e-grid .e-recordpluscollapse {
background-color: #deecf9;
}
`
```

In this example, the **.e-groupcaption** class targets the group caption row element, and the **.e-recordplusexpend** and **.e-recordpluscollapse** classes target the icons indicating record expansion or collapse. You can modify the **background-color** property to change the color of these elements.

Customer ID ↑ ×			
	ORDER ID	SHIP CITY	SHIP NAME
▼ Customer ID: CENTC - 1 item			
	10259	México D.F.	Centro comercial Moctezuma
▼ Customer ID: CHOPS - 1 item			
	10254	Bern	Chop-suey Chinese
▼ Customer ID: ERNSH - 1 item			
	10258	Graz	Ernst Handel
▼ Customer ID: HANAR - 2 items			
	10250	Rio de Janeiro	Hanari Carnes
	10253	Rio de Janeiro	Hanari Carnes

Customizing the grouping indent cell

To customize the appearance of the grouping indent cell element, you can use the following CSS code:

```
`css
.e-grid .e-indentcell {
background-color: #deecf9;
}
`
```

In this example, the **.e-indentcell** class targets the grouping indent cell element. You can modify the **background-color** property to change the color of the indent cell.

Customer ID ↑ ×			
	ORDER ID	SHIP CITY	SHIP NAME
▼	Customer ID: CENTC - 1 item		
	10259	México D.F.	Centro comercial Moctezuma
▼	Customer ID: CHOPS - 1 item		
	10254	Bern	Chop-suey Chinese
▼	Customer ID: ERNSH - 1 item		
	10258	Graz	Ernst Handel
▼	Customer ID: HANAR - 2 items		
	10250	Rio de Janeiro	Hanari Carnes
	10253	Rio de Janeiro	Hanari Carnes

Toolbar in EJ2 JavaScript Grid control



You can customize the appearance of the toolbar in the Syncfusion EJ2 JavaScript Grid control using CSS. Here are examples of how to customize the toolbar root element and toolbar button element.

Customizing the toolbar root element

To customize the appearance of toolbar root element, you can use the following CSS code:

```
`css
.e-grid .e-toolbar-items {
background-color: #deecf9;
}
```

In this example, the **.e-toolbar-items** class targets the background color of the toolbar root element. You can modify the **background-color** property to change the background color of the toolbar.


<div> <div>+ Add</div> <div> Update</div> <div> Cancel</div> </div>			
ORDER ID	CUSTOMER ID	SHIP CITY	SHIP NAME
10248	VINET	Reims	France
10249	TOMSP	Münster	Germany
10250	HANAR	Rio de Janeiro	Brazil
10251	VICTE	Lyon	France

Customizing the toolbar button element

To customize the appearance of toolbar buttons, you can use the following CSS code:

```
`css
.e-grid .e-toolbar .e-btn {
background-color: #deecf9;
}
```

In this example, the **.e-toolbar .e-btn** selector targets the background color of the toolbar button elements. You can modify the **background-color** property to change the background color of the toolbar buttons.

<div> <div>+ Add</div> <div> Update</div> <div> Cancel</div> </div>			
ORDER ID	CUSTOMER ID	SHIP CITY	SHIP NAME
10248	VINET	Reims	France
10249	TOMSP	Münster	Germany
10250	HANAR	Rio de Janeiro	Brazil
10251	VICTE	Lyon	France

Editing in EJ2 JavaScript Grid control

You can customize the appearance of editing-related elements in the Syncfusion EJ2 JavaScript Grid control using CSS. Below are examples of how to customize various editing-related elements.


Customizing the edited and added row element


To customize the appearance of edited and added row table elements in the grid, you can use the following CSS code:

```
`css
.e-grid .e-editedrow table, .e-grid .e-addedrow table {
```

```
background-color: #62b2eb;
}
```

In this example, the **.e-editedrow** class represents the edited row element, and the **.e-addedrow** class represents the added row element. You can modify the `background-color` property to change the color of these row table elements.

+ Add  Update X Cancel			
ORDER ID	CUSTOMER ID	SHIP CITY	SHIP NAME
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
10248	VINET	Reims	France
10249	TOMSP	Münster	Germany
10250	HANAR	Rio de Janeiro	Brazil
10251	VICTE	Lyon	France

+ Add  Update X Cancel			
ORDER ID	CUSTOMER ID	SHIP CITY	SHIP NAME
10248	VINET	Reims	France
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
10250	HANAR	Rio de Janeiro	Brazil
10251	VICTE	Lyon	France

Customizing the edited row input element

To customize the appearance of edited row input elements in the grid, you can use the following CSS code:

```
`css
.e-grid .e-gridform .e-rowcell .e-input-group .e-input.e-field {
font-family: cursive;
color:rgb(214, 33, 123)
}
```

In this example, the **.e-gridform** class represents the editing form, and the **.e-input** class represents the input elements within the form. You can modify the **font-family** property to change the font and **color** property to change text color of the input elements.

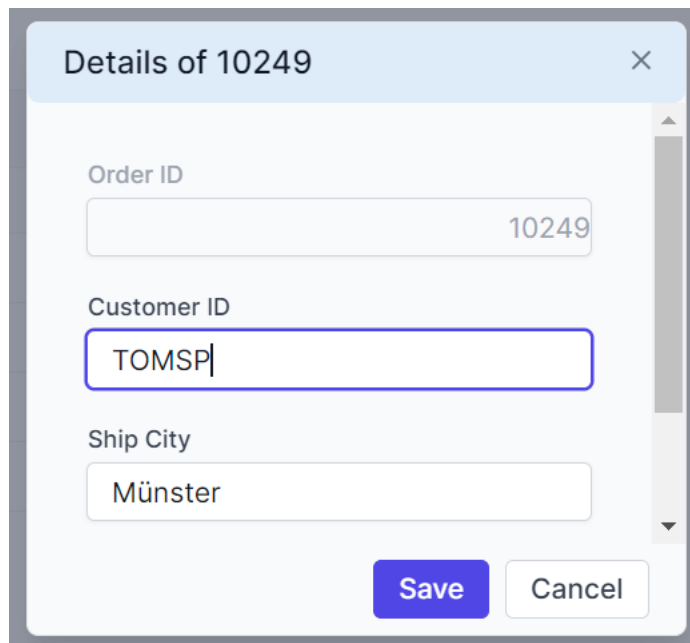
+ Add 📄 Update ✕ Cancel			
ORDER ID	CUSTOMER ID	SHIP CITY	SHIP NAME
10248	VINET	Reims	France
10249	TOMSP	Münster	Germany
10250	HANAR	Rio de Janeiro	Brazil
10251	VICTE	Lyon	France

Customizing the edit dialog header element

To customize the appearance of the edit dialog header element in the grid, you can use the following CSS code:

```
`css
.e-edit-dialog .e-dlg-header-content {
background-color: #deecf9;
}
`
```

In this example, the **.e-edit-dialog** class represents the edit dialog, and the **.e-dlg-header-content** class targets the header content within the dialog. You can modify the **background-color** property to change the color of the header element.



The image shows a dialog box titled "Details of 10249" with a close button (X) in the top right corner. Inside the dialog, there are three input fields: "Order ID" containing "10249", "Customer ID" containing "TOMSP", and "Ship City" containing "Münster". At the bottom of the dialog are two buttons: "Save" (highlighted in blue) and "Cancel".

Customizing the edited row input element in dialog edit mode

To customize the appearance of edited row input elements in dialog edit mode, you can use the following CSS code:

```
`css
.e-gridform .e-rowcell .e-float-input .e-field {
font-family: cursive;
}
`
```









In this example, the **.e-gridform** class represents the editing form, and the **.e-float-input** class targets the floating input elements within the form. You can modify the `font-family` property to change the font of the input elements.









Customizing the command column buttons

To customize the appearance of command column buttons such as edit, delete, update, and cancel, you can use the following CSS code:

```
`css
.e-grid .e-delete::before ,.e-grid .e-cancel-icon::before{
color: #f51717;
}
.e-grid .e-edit::before, .e-grid .e-update::before {
color: #077005;
}
`
```

In this example, the **.e-edit**, **.e-delete**, **.e-update**, and **.e-cancel-icon** classes represent the respective command column buttons. You can modify the `color` property to change the color of these buttons.

ORDER ID	CUSTOMER ID	FREIGHT	SHIP COUNTRY	COMMANDS	
10248	VINET	\$32.38	France		
10249	TOMSP	\$11.61	Germany		
10250	HANAR	\$65.83	Brazil		
10251	VICTE	\$41.34	France		

ORDER ID	CUSTOMER ID	FREIGHT	SHIP COUNTRY	COMMANDS	
10248	VINET	32....	France		
10249	TOMSP	\$11.61	Germany		
10250	HANAR	\$65.83	Brazil		
10251	VICTE	\$41.34	France		

Aggregate in EJ2 JavaScript Grid control

You can customize the appearance of aggregate elements in the Syncfusion EJ2 JavaScript Grid control using CSS. Below are examples of how to customize the aggregate root element and the aggregate cell elements.

Customizing the aggregate root element

To customize the appearance of the Grid's aggregate root elements, you can use the following CSS code:

```
`css
.e-grid .e-gridfooter {
font-family: cursive;
}
```

In this example, the **e-gridfooter** class represents the root element of the aggregate row in the grid footer. You can modify the **font-family** property to change the font of the aggregate root element.

Customer Name	Freight	Order Date	Ship Country
Paul Henriot	\$32.38	7/4/1996	France
Karin Josephs	\$11.61	7/5/1996	Germany
Mario Pontes	\$65.83	7/8/1996	Brazil
Mary Saveley	\$41.34	7/8/1996	France
Sum: \$64,942.69			
Average: \$78.24			

Customizing the aggregate cell elements

To customize the appearance of the Grid's aggregate cell elements (summary row cell elements), you can use the following CSS code:

```
`css
.e-grid .e-summaryrow .e-summarycell {
background-color: #deecf9;
}
```

In this example, the **e-summaryrow** class represents the summary row containing aggregate cells, and the **e-summarycell** class targets individual aggregate cells within the summary row. You can modify the **background-color** property to change the **color** of the aggregate cell elements.

Order ID	Customer ID	Freight	Ship Name
10248	VINET	32.38	Vins et alcools Chev...
10249	TOMSP	11.61	Toms Spezialitäten
10250	HANAR	65.83	Hanari Carnes
10251	VICTE	41.34	Victuailles en stock
		Sum: 151.16	
		Max: 65.83	

Selection in EJ2 JavaScript Grid control

You can customize the appearance of the selection in the Syncfusion EJ2 JavaScript Grid control using CSS. Here are examples of how to customize the row selection background, cell selection background, and column selection background.

Customizing the row selection background

To customize the appearance of row selection, you can use the following CSS code:

```
`css
.e-grid td.e-selectionbackground {
background-color: #00b7ea;
}
```

In this example, the **.e-selectionbackground** class targets the background color of the row selection. You can modify the **background-color** property to change the background color of the selected rows.

ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996

Customizing the cell selection background

To customize the appearance of cell selection, you can use the following CSS code:

```
`css
```



```
.e-grid td.e-cellselectionbackground {
background-color: #00b7ea;
}
`
```

In this example, the **.e-cellselectionbackground** class targets the background color of the cell selection. You can modify the **background-color** property to change the background color of the selected cells.

ORDER ID	CUSTOMER ID	FREIGHT	ORDER DATE
10248	VINET	\$32.38	7/4/1996
10249	TOMSP	\$11.61	7/5/1996
10250	HANAR	\$65.83	7/8/1996
10251	VICTE	\$41.34	7/8/1996

Customizing the column selection background

To customize the appearance of column selection, you can use the following CSS code:

```
`css
.e-grid .e-columnselection {
background-color: #aec2ec;
}
`
```

In this example, the **.e-columnselection** class targets the background color of the column selection. You can modify the **background-color** property to change the background color of the selected columns.

ORDER ID	CUSTOMER NAME	ORDER DATE	FREIGHT	SHIPPED DATE
10248	Paul Henriot	7/4/1996	\$32.38	7/16/1996
10249	Karin Josepchs	7/5/1996	\$11.61	7/10/1996
10250	Mario Pontes	7/8/1996	\$65.83	7/12/1996
10251	Mary Saveley	7/8/1996	\$41.34	7/15/1996
10252	Pascale Cartrain	7/9/1996	\$51.30	7/11/1996

Migration in EJ2 JavaScript Grid control

The following enum property values are changed from camel casing to pascal casing.

Enum Name | Old Property Value | New Property Value

SelectionMode | single | Single

SelectionType | multiple | Multiple
SelectionMode | cell | Cell
SelectionMode | row | Row
SelectionMode | both | Both
CellSelectionMode | flow | Flow
CellSelectionMode | box | Box
EditMode | normal | Normal
EditMode | dialog | Dialog
EditMode | batch | Batch
TextAlign | left | Left
TextAlign | right | Right
TextAlign | center | Center
TextAlign | justify | Justify
GridLine | both | Both
GridLine | none | None
GridLine | horizontal | Horizontal
GridLine | vertical | Vertical
GridLine | default | Default
PrintMode | allPages | AllPages
PrintMode | currentPage | CurrentPage
FilterType | filterbar | FilterBar
FilterType | excel | Excel
FilterType | menu | Menu
FilterType | checkbox | CheckBox
FilterBarMode | onenter | OnEnter
FilterBarMode | immediate | Immediate
AggregateType | sum | Sum
AggregateType | average | Average
AggregateType | max | Max
AggregateType | min | Min
AggregateType | count | Count
AggregateType | truecount | TrueCount
AggregateType | falsecount | FalseCount

AggregateType | custom | Custom
WrapMode | both | Both
WrapMode | header | Header
WrapMode | content | Content
MultipleExportType | appendtosheet | AppendToSheet
MultipleExportType | newsheet | NewSheet
ToolBarItems | add | Add
ToolBarItems | delete | Delete
ToolBarItems | update | Update
ToolBarItems | cancel | Cancel
ToolBarItems | edit | Edit
ToolBarItems | search | Search
ToolBarItems | columnchooser | ColumnChooser
ToolBarItems | print | Print
ToolBarItems | pdfexport | PdfExport
ToolBarItems | excelexport | ExcelExport
ToolBarItems | csvexport | CsvExport
ToolBarItems | wordexport | WordExport
ClipMode | clip | Clip
ClipMode | ellipsis | Ellipsis
ClipMode | ellipsiswithtooltip | EllipsisWithTooltip
CommandButtonType | edit | Edit
CommandButtonType | delete | Delete
CommandButtonType | save | Save
CommandButtonType | cancel | Cancel
ContextMenuItem | autoFitAll | AutoFitAll
ContextMenuItem | autoFit | AutoFit
ContextMenuItem | group | Group
ContextMenuItem | ungroup | Ungroup
ContextMenuItem | edit | Edit
ContextMenuItem | delete | Delete
ContextMenuItem | save | Save
ContextMenuItem | cancel | Cancel

ContextMenuItem |copy |Copy
ContextMenuItem |pdfExport |PdfExport
ContextMenuItem |excelExport |ExcelExport
ContextMenuItem |csvExport |CsvExport
ContextMenuItem |sortAscending |SortAscending
ContextMenuItem |sortDescending |SortDescending
ContextMenuItem |firstPage |FirstPage
ContextMenuItem |prevPage |PrevPage
ContextMenuItem |lastPage |LastPage
ContextMenuItem |nextPage |NextPage
ColumnMenuItem |autoFitAll |AutoFitAll
ColumnMenuItem |autoFit |AutoFit
ColumnMenuItem |group |Group
ColumnMenuItem |ungroup |Ungroup
ColumnMenuItem |sortAscending |SortAscending
ColumnMenuItem |sortDescending |SortDescending
ColumnMenuItem |columnChooser |ColumnChooser
ColumnMenuItem |filter |Filter
PdfPageSize |letter |Letter
PdfPageSize |note |Note
PdfPageSize |legal |Legal
PdfPageSize |a0 |A0
PdfPageSize |a1 |A1
PdfPageSize |a2 |A2
PdfPageSize |a3 |A3
PdfPageSize |a4 |A4
PdfPageSize |a5 |A5
PdfPageSize |a6 |A6
PdfPageSize |a7 |A7
PdfPageSize |a8 |A8
PdfPageSize |a9 |A9
PdfPageSize |b0 |B0
PdfPageSize |b1 |B1

PdfPageSize |b2 |B2
PdfPageSize |b3 |B3
PdfPageSize |b4 |B4
PdfPageSize |b5 |B5
PdfPageSize |archa |Archa
PdfPageSize |archb |Archb
PdfPageSize |archc |Archc
PdfPageSize |archd |Archd
PdfPageSize |arche |Arche
PdfPageSize |flsa |Flsa
PdfPageSize |halfletter |HalfLetter
PdfPageSize |letter11x17 |Letter11x17
PdfPageSize |ledger |Ledger
PageOrientation |landscape |Landscape
PageOrientation |portrait |Portrait
ContentType |image |Image
ContentType |line |Line
ContentType |pagenumber |PageNumber
ContentType |text |Text
PdfPageNumberType |lowerlatin |LowerLatin
PdfPageNumberType |lowerroman |LowerRoman
PdfPageNumberType |upperlatin |UpperLatin
PdfPageNumberType |upperroman |UpperRoman
PdfPageNumberType |numeric |Numeric
PdfPageNumberType |arabic |Arabic
PdfDashStyle |solid |Solid
PdfDashStyle |dash |Dash
PdfDashStyle |dot |Dot
PdfDashStyle |dashdot |DashDot
PdfDashStyle |dashdotdot |DashDotDot
PdfHAlign |left |Left
PdfHAlign |right |Right
PdfHAlign |center |Center

PdfHAlign |justify |Justify

PdfVAlign |top |Top

PdfVAlign |bottom |Bottom

PdfVAlign |middle |Middle

ExportType |currentpage |CurrentPage

ExportType |allpages |AllPages

ExcelHAlign |left |Left

ExcelHAlign |right |Right

ExcelHAlign |center |Center

ExcelHAlign |fill |Fill

ExcelVAlign |top |Top

ExcelVAlign |bottom |Bottom

ExcelVAlign |center |Center

ExcelVAlign |justify |Justify

BorderLineStyle |thin |Thin

BorderLineStyle |thick |Thick

[Ej1 api migration in EJ2 JavaScript Grid control](#)

This article describes the API migration process of Grid component from Essential JS 1 to Essential JS 2.

Sorting

|Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

|Default | **Property:** *allowSorting*

 `$("#Grid").ejGrid({
 allowSorting: true
});` | **Property:** *allowSorting*

 `var gridObj = new ej.grids.Grid({ allowSorting: true
});
gridObj.appendTo('#Grid');` |

|Clear the Sorted columns | **Method:** *clearSorting()*

 `var gridObj = $("#Grid").data("ejGrid");
gridObj.clearSorting();` | **Method:** *clearSorting()*

 `var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.clearSorting()`

|Get the Sorted Columns by using the Fieldname | **Method:** *getsortColumnByField(field)*

 `var gridObj = $("#Grid").data("ejGrid");
gridObj.getsortColumnByField("OrderID");` | **Property:** *sortSettings.columns*

 Sorted Column for a particular field can be get by iterating the *sortSettings.columns* with the fieldname

|Remove the Sorted Columns | **Method:** *removeSortedColumns(fieldName)*

 `var gridObj = $("#Grid").data("ejGrid");
gridObj.removeSortedColumns("OrderID");` | **Method:** *removeSortColumn()*

 `var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.removeSortColumn("OrderID")`

| Sort a Column by using the method | **Method:** *sortColumn(columnName, [sortingDirection])*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.sortColumn("OrderID", "ascending"); |
Method: *sortColumn(columnName, Direction)*

 var gridObj =
 document.getElementById('Grid')
.ej2_instances[0]
 gridObj.sortColumn("OrderID",
 "ascending");

Grouping

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *allowGrouping*

 \$("#Grid").ejGrid({
 allowGrouping: true

 }); | **Property:** *allowGrouping*

 var gridObj = new
 ej.grids.Grid({
 allowGrouping: true
});
gridObj.appendTo('#Grid'); |

| Group Columns initially | **Property:** *groupSettings.groupedColumns*

 \$("#Grid").ejGrid({
 allowGrouping: true,
 groupSettings: {

 groupedColumns:["OrderID", "CustomerID"]
 }
}); | **Property:**
groupSettings.columns

 var gridObj = new ej.grids.Grid({
 allowGrouping:
 true,
 groupSettings: {
 columns: ["OrderID",
 "CustomerID"]
 }
});
gridObj.appendTo('#Grid');

| Caption Template | **Property:** *groupSettings.captionFormat*

 \$("#Grid").ejGrid({
 allowGrouping: true,
 groupSettings: {

 captionFormat: "#template"
 }
}); | **Property:**
groupSettings.captionTemplate

 var gridObj = new
 ej.grids.Grid({
 allowGrouping: true,
 groupSettings:
 {
 captionTemplate:
 '#template'
 }
});
gridObj.appendTo('#Grid');

| Show Drop Area | **Property:** *groupSettings.showDropArea*

 \$("#Grid").ejGrid({
 allowGrouping: true,
 groupSettings: {

 showDropArea: false
 }
}); | **Property:**
groupSettings.showDropArea

 var gridObj = new ej.grids.Grid({
 allowGrouping:
 true,
 groupSettings: {
 showDropArea:
 false
 }
});
gridObj.appendTo('#Grid');

| Collapse all group caption rows | **Method:** *collapseAll()*

 var gridObj =
 \$("#Grid").data("ejGrid");
 gridObj.collapseAll(); | **Method:** *collapseAll()*

 var gridObj =
 document.getElementById('Grid')
.ej2_instances[0]
 gridObj.groupModule.collapseAll()

| Expand all group caption rows | **Method:** *expandAll()*

 var gridObj =
 \$("#Grid").data("ejGrid");
 gridObj.expandAll(); | **Method:** *expandAll()*

 var gridObj =
 document.getElementById('Grid')
.ej2_instances[0]
 gridObj.groupModule.expandAll()

| Expand or collapse the row based
on the row state in grid | **Method:** *expandCollapse(\$target)*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.expandCollapse(\$("#tr
 td.recordplusexpend > div").first()); | **Method:** *expandCollapseRows()*

 var gridObj =
 document.getElementById('Grid')
.ej2_instances[0]
gridObj.groupModule.expandCollap
 seRows(
gridObj.getContent().querySelectorAll('.e-recordplusexpend')[0])

| Collapse the group drop area in grid | **Method:** *collapseGroupDropArea()*

 `var gridObj = $("#Grid").data("ejGrid");`
 `gridObj.collapseGroupDropArea();` | Not Applicable

| Expand the group drop area in grid | **Method:** *expandGroupDropArea()*

 `var gridObj = $("#Grid").data("ejGrid");`
 `gridObj.expandGroupDropArea();` | Not Applicable

| Group a column by using the method | **Method:** *groupByColumn(fieldName)*

 `var gridObj = $("#Grid").data("ejGrid");`
 `gridObj.groupColumn("OrderID");` | **Method:** *groupByColumn(fieldName)*

 `var gridObj = document.getElementById('Grid')`
 `.ej2_instances[0]`
 `gridObj.groupColumn("OrderID")`

| Ungroup a grouped column by using the method | **Method:** *ungroupColumn(fieldName)*

 `var gridObj = $("#Grid").data("ejGrid");`
 `gridObj.ungroupColumn("OrderID");` | **Method:** *ungroupColumn(fieldName)*

 `var gridObj = document.getElementById('Grid')`
 `.ej2_instances[0]`
 `gridObj.ungroupColumn("OrderID")`

Filtering

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| ----- | ----- | ----- |

| Default | **Property:** *allowFiltering*

 `$("#Grid").ejGrid({
 allowFiltering: true
});` | **Property:** *allowFiltering*

 `var gridObj = new ej.grids.Grid({
 allowFiltering: true
});`
 `gridObj.appendTo('#Grid');` |

| Menu Filtering | **Property:** *filterSettings.filterType*

 `$("#Grid").ejGrid({
 allowFiltering: true,
 filterSettings: {
 filterType : "menu"
 }
});` | **Property:** *filterSettings.type*

 `var gridObj = new ej.grids.Grid({
 allowFiltering: true,
 filterSettings: {
 type:'Menu'
 }
});`
 `gridObj.appendTo('#Grid');`

| Excel Filtering | **Property:** *filterSettings.filterType*

 `$("#Grid").ejGrid({
 allowFiltering: true,
 filterSettings: {
 filterType : "excel"
 }
});` | **Property:** *filterSettings.type*

 `var gridObj = new ej.grids.Grid({
 allowFiltering: true,
 filterSettings: {
 type:'Excel'
 }
});`
 `gridObj.appendTo('#Grid');`

| Clear the Filtered values | **Method:** *clearFiltering(field)* - *field is optional*

 `var gridObj = $("#Grid").data("ejGrid");`
 `gridObj.clearFiltering();` | **Method:** *clearFiltering()*

 `var gridObj = document.getElementById('Grid')`
 `.ej2_instances[0]`
 `gridObj.clearFiltering()`

| Filter a column by using the method | **Method:** *filterColumn(fieldName, filterOperator, filterValue,
predicate, [matchcase],[actualFilterValue])*

 `var gridObj = $("#Grid").data("ejGrid");`
 `gridObj.filterColumn("OrderID",
"equal","10248","and", true);` | **Method:** *filterByColumn(fieldName, filterOperator, filterValue, predicate, matchCase, ignoreAccent, actualFilterValue, actualOperator)*

 `var gridObj = document.getElementById('Grid')`
 `.ej2_instances[0]`
 `gridObj.filterByColumn("OrderID","equal",10248)`

| Filter columns by Collection | **Method:** *filterColumn(filterCollection)*

 `var gridObj = $("#Grid").data("ejGrid");`


```
gridObj.filterColumn([{field:"OrderID",<br>&#160;operator:"lessthan",value:"10266",<br>predic
ate:"and",matchcase:true},<br>&#160;{field:"EmployeeID",operator:<br>&#160;"equal",value:2,
predicate:"and", matchcase:true}])| Property: filterSettings.columns <br><br> var gridObj =
document.getElementById('Grid')<br>.ej2_instances[0] <br>gridObj.filterSettings:{columns: [{
field: 'ShipCity', matchCase: false, operator: 'startswith', predicate: 'and', value: 'reims' ]}}
```

| Get the Filtered Records | **Method:** `getFilteredRecords()`

 `var gridObj = $(“#Grid”).data(“ejGrid”);
 gridObj.getFilteredRecords();` | Not Applicable

Searching

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

```
| Default | Property: toolbarSettings.toolbarItems <br><br>
$(“#Grid”).ejGrid({<br>&#160;&#160;showToolBar: true, toolbarItems : [“search”]<br>
&#160;&#160;allowSearching : true<br>}); | Property: toolbar <br><br> var gridObj = new
ej.grids.Grid({<br>&#160;toolbar: [‘Search’]<br>});<br>gridObj.appendTo(‘#Grid’);
```

```
| Clear the Searched values | Method: clearSearching() <br><br> var gridObj =  
$(("#Grid").data("ejGrid")); <br> gridObj.clearSearching(); | Method: searchModule.search() <br><br>  
var gridObj = document.getElementById('Grid')<br>.ej2_instances[0] <br>  
gridObj.searchModule.search("");
```

```
| Search a value | Method: search(searchString) <br><br> var gridObj = $("#Grid").data("ejGrid");  
<br> gridObj.search("France"); | Method: searchModule.search() <br><br> var gridObj =  
document.getElementById('Grid')<br>.ej2_instances[0] <br>  
gridObj.searchModule.search("France");
```

Paging

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

```
| Default | Property: allowPaging <br><br> $(" #Grid").ejGrid({ <br>&#160; allowPaging: true <br>}); | Property: allowPaging <br><br>var gridObj = new ej.grids.Grid({ &#160;allowPaging: true <br>});<br>gridObj.appendTo('#Grid'); |
```

| Customize Paging | **Property:** *pageSettings.pageSize*

```
<br><br>$("#Grid").ejGrid({<br>&#160;&#160;allowPaging: true,<br>&#160;pageSettings: {<br>&#160;&#160;pageSize: 8,pageSizeList: [5, 10],pageCount:3<br>&#160;&#160;}<br>});|Property:  
pageSettings.pageSize <br><br> var gridObj = new ej.grids.Grid({<br>&#160;allowPaging:  
true,<br>&#160;pageSettings: {<br>&#160;pageSize:8, pageSize: [5,  
10],pageCount:3<br>&#160;&#160;}<br>});<br>gridObj.appendTo('#Grid');
```

| Change Page Size | **Method:** `changePageSize(pageSize)`

 `var gridObj = $("#Grid").data("ejGrid");`
 `gridObj.changePageSize(7);` | **Property:** `pageSettings.pageSize`

 Pagesize can be modified dynamically by using the below code
 `pageSettings.pageSize = 7;` |

| Get Current Page Index | **Method:** *getCurrentIndex()*

 var gridObj =
 \$("#Grid").data("ejGrid");
 gridObj.getCurrentIndex(); | **Property:** *pageSettings.currentPage*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]

 gridObj.pageSettings.currentPage;

| Get Pager Element | **Method:** *getPager()*

 var gridObj = \$("#Grid").data("ejGrid");

 gridObj.getPager(); | **Method:** *getPager()*

 var gridObj =
 document.getElementById('Grid')
.ej2_instances[0]
 gridObj.getPager();

| Send a paging request to specified Page | **Method:** *gotoPage(pageIndex)*

 var gridObj =
 \$("#Grid").data("ejGrid");
 gridObj.gotoPage(3); | **Method:** *gotoPage(pageIndex)*

 var
 gridObj = document.getElementById('Grid')
.ej2_instances[0]
 gridObj.gotoPage(3);

| Calculate Pagesize of grid by using its Parent height(containerHeight) | **Method:**
calculatePageSizeByParentHeight(containerHeight)

 var gridObj =
 \$("#Grid").data("ejGrid");
 gridObj.
calculatePageSizeByParentHeight(400); | Not
 Applicable

Selection

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *allowSelection*

 \$("#Grid").ejGrid({
 allowSelection: true

}); | **Property:** *allowSelection*

 var gridObj = new ej.grids.Grid({ allowSelection:
 true
});
gridObj.appendTo('#Grid'); |

| Single Selection | **Property:**
selectionType

\$("#Grid").ejGrid({
 allowSelection:
 true,
 selectionType : "single"
 }); | **Property:**
selectionSettings.type

var gridObj = new ej.grids.Grid({
 allowSelection:
 true,
 selectionSettings: {
 type :
 "Single"
 }
});
gridObj.appendTo('#Grid');

| Multiple Selection | **Property:** *selectionType*

 \$("#Grid").ejGrid({
 allowSelection: true,
 selectionType :
 "multiple"
 }); | **Property:** *selectionSettings.type*

var gridObj = new
 ej.grids.Grid({
 allowSelection: true,
 selectionSettings: {
 type :
 "Multiple"
 }
});
gridObj.appendTo('#Grid');

| Row Selection | **Property:** *selectionSettings.selectionMode*

 \$("#Grid").ejGrid({
 allowSelection: true,
 selectionSettings :
 {selectionMode: ["row"]
 }); | **Property:** *selectionSettings.mode*

var
 gridObj = new ej.grids.Grid({
 allowSelection: true,
 selectionSettings:
 {
 mode: 'Row'
 }
});
gridObj.appendTo('#Grid');

| Cell Selection | **Property:** *selectionSettings.selectionMode*

 \$("#Grid").ejGrid({
 allowSelection: true,
 selectionSettings :
 {selectionMode: ["cell"]
 }); | **Property:** *selectionSettings.mode*

var

```
gridObj = new ej.grids.Grid({  
allowSelection: true,  
selectionSettings:  
mode: 'Cell'  
});  
gridObj.appendTo("#Grid");
```

| Clear the selected Cells | **Method:** *clearCellSelection()*

var gridObj = \$("#Grid").data("ejGrid");
gridObj.clearCellSelection(); | **Method:** *clearCellSelection()*

var gridObj = document.getElementById('Grid').ej2_instances[0]
gridObj.selectionModule.clearCellSelection();

| Clear the selected Columns | **Method:** *clearColumnSelection([index])* - index is optional

var gridObj = \$("#Grid").data("ejGrid");
gridObj.clearColumnSelection(); | Not Applicable

| Get the selected Records | **Method:** *getSelectedRecords()*

var gridObj = \$("#Grid").data("ejGrid");
gridObj.getSelectedRecords(); | **Method:** *getSelectedRecords()*

var gridObj = document.getElementById('Grid').ej2_instances[0]
gridObj.getSelectedRecords();

| Get the selected Rows | **Method:** *getSelectedRows()*

var gridObj = \$("#Grid").data("ejGrid");
gridObj.getSelectedRows(); | **Method:** *getSelectedRows()*

var gridObj = document.getElementById('Grid').ej2_instances[0]
gridObj.getSelectedRows();

| Select Cells | **Method:** *selectCells(rowCellIndexes)*

var gridObj = \$("#Grid").data("ejGrid");
gridObj.selectCells([[1, [4, 3, 2]]]); | **Method:** *selectionModule.selectCells();*

var gridObj = document.getElementById('Grid').ej2_instances[0]
gridObj.selectionModule.selectCells([{ rowIndex: 0, cellIndexes: [0] }, { rowIndex: 1, cellIndexes: [1] }]);

| Select Rows | **Method:** *selectRows(fromIndex, toIndex)*

var gridObj = \$("#Grid").data("ejGrid");
gridObj.selectRows(1, 4); | **Method:** *selectionModule.selectRows()*

var gridObj = document.getElementById('Grid').ej2_instances[0]
gridObj.selectionModule.selectRows([0, 2])

| Triggers when a cell is selected | **Event:** *cellSelected*

\$("#Grid").ejGrid({
cellSelected: function (args) {}
}); | **Event:** *cellSelected*

var gridObj = new ej.grids.Grid({
cellSelected: function (args) {}
});

| Triggers before the cell is being selected | **Event:** *cellSelecting*

\$("#Grid").ejGrid({
cellSelecting: function (args) {}
}); | **Event:** *cellSelecting*

var gridObj = new ej.grids.Grid({
cellSelecting: function (args) {}
});

| Triggers when a cell is deselected | **Event:** *cellDeselected*

\$("#Grid").ejGrid({
cellDeselected: function (args) {}
}); | **Event:** *cellDeselected*

var gridObj = new ej.grids.Grid({
cellDeselected: function (args) {}
});

| Triggers before the cell is being deselected | **Event:** *cellDeselecting*

\$("#Grid").ejGrid({
cellDeselecting: function (args) {}
}); | **Event:** *cellDeselecting*

var gridObj = new ej.grids.Grid({
cellDeselecting: function (args) {}
});

| Triggers when the row is selected | **Event:** *rowSelected*
 `$("#Grid").ejGrid({
rowSelected: function (args){
ej.grids.Grid({
rowSelected: function (args){` }
}); | **Event:** *rowSelected*
 `var gridObj = new`

| Triggers before the row is being selected | **Event:** *rowSelecting*
 `$("#Grid").ejGrid({
rowSelecting: function (args){
}); | Event: rowSelecting
 var gridObj = new ej.grids.Grid({
rowSelecting: function (args){ }
});`

| Triggers when the row is deselected | **Event:** *rowDeselected*
 `$("#Grid").ejGrid({
rowDeselected: function (args){
}); | Event: rowDeselected
 var gridObj = new ej.grids.Grid({
rowDeselected: function (args){ }
});`

| Triggers before the row is being deselected | **Event:** *rowDeselecting*
 `$("#Grid").ejGrid({
rowDeselecting: function (args){
}); | Event: rowDeselecting
 var gridObj = new ej.grids.Grid({
rowDeselecting: function (args){ }
});`

| Triggers when the column is selected | **Event:** *columnSelected*
 `$("#Grid").ejGrid({
columnSelected: function (args){
}); | Not Applicable`

| Triggers before the column is being selected | **Event:** *columnSelecting*
 `$("#Grid").ejGrid({
columnSelecting: function (args){
}); | Not Applicable`

| Triggers when the column is deselected | **Event:** *columnDeselected*
 `$("#Grid").ejGrid({
columnDeselected: function (args){
}); | Not Applicable`

| Triggers before the column is being deselected | **Event:** *columnDeselecting*
 `$("#Grid").ejGrid({
columnDeselecting: function (args){
}); | Not Applicable`

Editing

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *editSettings*
 `$("#Grid").ejGrid({
 editSettings : { allowEditing: true,
 allowAdding: true, allowDeleting: true }
}); | Property: editSettings
 var gridObj = new ej.grids.Grid({
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true }
});
gridObj.appendTo('#Grid');`

| Inline Editing | **Property:** *editSettings.editMode*
 `$("#Grid").ejGrid({
 editSettings : { allowEditing: true,
 allowAdding: true, allowDeleting: true, editMode : "normal" }
}); | Property: editSettings.mode
 var gridObj = new ej.grids.Grid({
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Normal' }
});
gridObj.appendTo('#Grid');`

| Dialog Editing | **Property:** *editSettings.editMode*
 `$("#Grid").ejGrid({
 editSettings : { allowEditing: true,
 allowAdding: true, allowDeleting: true, editMode : "dialog" }
}); | Property: editSettings.mode
 var gridObj = new ej.grids.Grid({
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Dialog' }
});
gridObj.appendTo('#Grid');`

| Batch Editing | **Property:** *editSettings.editMode*
 `$("#Grid").ejGrid({
 editSettings : { allowEditing: true,
 allowAdding: true,`

allowDeleting: true, editMode : "batch">}}; | **Property:** *editSettings.mode*

 var gridObj = new ej.grids.Grid({
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Batch' }
});
gridObj.appendTo('#Grid');

| Add a new Record | **Method:** *addRecord([data,[serverChange]])*

 var gridObj = \$('#Grid').data("ejGrid");
 gridObj.addRecord({ "OrderID":12333}); | **Method:** *addRecord(data(optional), index(optional))*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
 gridObj.addRecord();

| Batch Cancel | **Method:** *batchCancel()*

 var gridObj = \$('#Grid').data("ejGrid");
 gridObj.batchCancel(); | Not Applicable

| Batch Save | **Method:** *batchSave()*

 var gridObj = \$('#Grid').data("ejGrid");
 gridObj.batchSave(); | **Method:** *batchSave()*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
 gridObj.editModule.batchSave()

| Save a Cell - If *preventSaveEvent* is true, then it
will prevent the client side cellSave event | **Method:** *saveCell([preventSaveEvent])*

 var gridObj = \$('#Grid').data("ejGrid");
 gridObj.saveCell(); | **Method:** *saveCell()*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
 gridObj.editModule.saveCell()

| End Edit | **Method:** *endEdit()*

 var gridObj = \$('#Grid').data("ejGrid");
 gridObj.endEdit(); | **Method:** *endEdit()*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
 gridObj.endEdit()

| Cancel Edit | **Method:** *cancelEdit()*

 var gridObj = \$('#Grid').data("ejGrid");
 gridObj.cancelEdit(); | **Method:** *closeEdit()*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
 gridObj.closeEdit()

| Delete Record | **Method:** *deleteRecord(fieldName, data)*

 var gridObj = \$('#Grid').data("ejGrid");
 gridObj.deleteRecord("OrderID", { OrderID: 10249, EmployeeID: 3 }); | **Method:** *deleteRecord(field, data)*- *Field and data are optional*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
 gridObj.deleteRecord()

| Delete Row | **Method:** *deleteRow(\$tr)*

 var gridObj = \$('#Grid').data("ejGrid");
 gridObj.deleteRow(\$(".gridcontent tr").first()); | **Method:** *deleteRow(tr)*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
 grid.deleteRow(grid.getContentTable()
.querySelector("tbody").firstChild)

| Edit a cell in Batch edit mode | **Method:** *editCell(index, fieldName)*

 var gridObj = \$('#Grid').data("ejGrid");
 gridObj.editCell(2, "OrderID"); | **Method:** *startEdit(tr)*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.editModule.editCell(0,gridObj
.columns[0].field)

| Edit Form Validation | **Method:** *editFormValidate()*

 var gridObj = \$('#Grid').data("ejGrid");
 gridObj.editFormValidate(); | **Method:** *editModule.formObj.validate()*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.editModule.formObj.validate()

|Get Batch Changes | **Method:** *getBatchChanges()*

 var gridObj = \$("#Grid").data("ejGrid");

 gridObj.getBatchChanges(); | **Method:** *editModule.getBatchChanges()*

 var gridObj =
 document.getElementById('Grid')
.ej2_instances[0]

gridObj.editModule.getBatchChanges()

|Refresh Batch Edit Changes | **Method:** *refreshBatchEditChanges()*

 var gridObj =
 \$("#Grid").data("ejGrid");
 gridObj.refreshBatchEditChanges(); | Not Applicable

|Set Default Data for adding| **Method:** *setDefaultData(defaultData)*

 var gridObj =
 \$("#Grid").data("ejGrid");
 var defaultData = {OrderID:"10000"};

 gridObj.setDefaultData(defaultData); | **Method:** *editModule.addRecord()*

 var gridObj =
 document.getElementById('Grid')
.ej2_instances[0]

 gridObj.editModule.addRecord({OrderID:10000})

|Start Edit | **Method:** *startEdit(\$tr)*

 var gridObj = \$("#Grid").data("ejGrid");

 gridObj.startEdit(\$(".gridcontent tr").first()); | **Method:** *startEdit()*

 var gridObj =
 document.getElementById('Grid')
.ej2_instances[0]

 gridObj.startEdit(gridObj.selectRow(0))

|Update Record | **Method:** *updateRecord(fieldName, data)*

 var gridObj =
 \$("#Grid").data("ejGrid");
 gridObj.updateRecord("OrderID", { OrderID: 10249, EmployeeID:
 3 }); | Not Applicable

|Set Cell value | **Method:** *setCellText()*

 var gridObj = \$("#Grid").data("ejGrid");

 gridObj.setCellText(0, 1, "France"); | **Method:** *setCellValue(key, field, value)*

 var gridObj =
 document.getElementById('Grid')
.ej2_instances[0]

 gridObj.setCellValue(10248,"CustomerID","A")

|Get Currently edited cell value| **Method:** *getCurrentEditCellData()*

 var gridObj =
 \$("#Grid").data("ejGrid");
 gridObj.getCurrentEditCellData(); | **Method:**
getCurrentEditCellData()

 var gridObj =
 document.getElementById('Grid')
.ej2_instances[0]

 gridObj.
editModule.getCurrentEditCellData();

|Triggers when adding a
record in batch editing| **Event:** *batchAdd*

 \$("#Grid").ejGrid({

 batchAdd: function (args){}
 }); | **Event:** *batchAdd*

 var gridObj = new
 ej.grids.Grid({
 batchAdd: function (args){}
 });

|Triggers when deleting a
record in batch editing| **Event:** *batchDelete*

 \$("#Grid").ejGrid({
 batchDelete: function (args){}
 }); | **Event:** *batchDelete*

 var gridObj = new ej.grids.Grid({
 batchDelete: function (args){}
 });

|Triggers before adding a record in batch editing| **Event:** *beforeBatchAdd*

 \$("#Grid").ejGrid({

 beforeBatchAdd: function (args){}
 }); | **Event:** *beforeBatchAdd*

 var gridObj
 = new ej.grids.Grid({
 beforeBatchAdd: function (args){}
 });

|Triggers before deleting a record in batch editing| **Event:** *beforeBatchDelete*

 \$("#Grid").ejGrid({
 beforeBatchDelete: function (args){}
 }); | **Event:**
beforeBatchDelete

 var gridObj = new ej.grids.Grid({
 beforeBatchDelete:
 function (args){}
 });

|Triggers before saving a record in batch editing| **Event:** *beforeBatchSave*

 `$("#Grid").ejGrid({

#160; beforeBatchSave: function (args){}
 });`| **Event:** *beforeBatchSave*

 `var
gridObj = new ej.grids.Grid({
#160; beforeBatchSave: function (args){}
 });`;

|Triggers before the
record in being edited| **Event:** *beginEdit*

 `$("#Grid").ejGrid({

#160; beginEdit: function (args){}
 });`| **Event:** *beginEdit*

 `var gridObj = new
ej.grids.Grid({
#160; beginEdit: function (args){}
 });`;

|Triggers when the
cell is being edited| **Event:** *cellEdit*

 `$("#Grid").ejGrid({
#160;
cellEdit: function (args){}
 });`| **Event:** *cellEdit*

 `var gridObj = new ej.grids.Grid({

#160; cellEdit: function (args){}
 });`;

|Triggers when the
cell is saved| **Event:** *cellSave*

 `$("#Grid").ejGrid({
#160;
cellSave: function (args){}
 });`| **Event:** *cellSave*

 `var gridObj = new ej.grids.Grid({

#160; cellSave: function (args){}
 });`;

|Triggers when the record is added| **Event:** *endAdd*

 `$("#Grid").ejGrid({
#160; endAdd:
function (args){}
 });`| **Event:** *actionComplete*

 `var gridObj = new ej.grids.Grid({

#160; actionComplete: function (args){}
 });`;

|Triggers when the record is deleted| **Event:** *endDelete*

 `$("#Grid").ejGrid({
#160;
endDelete: function (args){}
 });`| **Event:** *actionComplete*

 `var gridObj = new
ej.grids.Grid({
#160; actionComplete: function (args){}
 });`;

|Triggers after the record is edited| **Event:** *endEdit*

 `$("#Grid").ejGrid({
#160; endEdit:
function (args){}
 });`| **Event:** *actionComplete*

 `var gridObj = new ej.grids.Grid({

#160; actionComplete: function (args){}
 });`;

Resizing

|Behavior | API in Essential JS 1 | API in Essential JS 2 |

|----- | ----- | ----- |

|Default | **Property:** *allowResizing*

 `$("#Grid").ejGrid({
#160; allowResizing: true

});`| **Property:** *allowResizing*

 `var gridObj = new ej.grids.Grid({ allowResizing: true

});`
`gridObj.appendTo("#Grid");` |

|Resize a column by using the method| **Method:** *resizeColumns(column,width)*

 `var gridObj =
$("#Grid").data("ejGrid");`
 `gridObj.resizeColumns("OrderID",width);`| **Property:**
columns.width

To resize a column, set width to that particular column and then refresh the grid
by using the refresh method.
`var gridObj =
document.getElementById('Grid')`
`.ej2_instances[0]`
 `gridObj.columns[1].width =
100;`
`gridObj.refresh();`

|Triggers when a column resize starts| **Event:** *resizeStart*

 `$("#Grid").ejGrid({
#160;
resizeStart: function (args){}
 });`| **Event:** *resizeStart*

 `var gridObj = new ej.grids.Grid({

#160; resizeStart: function (args){}
 });`;

|Triggers when a column is resized| **Event:** *resized*

 `$("#Grid").ejGrid({
#160; resized:
function (args){}
 });`| **Event:** *resizeStop*

 `var gridObj = new ej.grids.Grid({
#160;
resizeStop: function (args){}
 });`;

| Triggers when a column resize stops | **Event:** *resizeEnd*

 `$("#Grid").ejGrid({

resizeEnd: function (args){}
 });` | Not Applicable

Reordering

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *allowReordering*

 `$("#Grid").ejGrid({
 allowReordering:
true
 });` | **Property:** *allowReordering*

 `var gridObj = new ej.grids.Grid({
 allowReordering: true
});
 gridObj.appendTo('#Grid');` |

| Reorder Columns | **Method:** *reorderColumns(fromFieldName, toFieldName)*

 `var gridObj =
$("#Grid").data("ejGrid");
 gridObj.reorderColumns("OrderID", "CustomerID");` | **Method:**
reorderColumns(fromFieldName, toFieldName)

 `var gridObj =
document.getElementById('Grid')
 .ej2_instances[0]
 gridObj.reorderColumns("OrderID",
"CustomerID");`

| Reorder Rows | **Method:** *reorderRows(indexes, toIndex)*

 `var gridObj =
$("#Grid").data("ejGrid");
 gridObj.reorderRows([0,1],3);` | Not Applicable

Context Menu

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *contextMenuSettings.enableContextMenu*

 `$("#Grid").ejGrid({
 contextMenuSettings: { enableContextMenu: true }
 });`
| **Property:** *contextMenuItems*

 `var gridObj = new
ej.grids.Grid({
 contextMenuItems: ['AutoFit',
'AutoFitAll']
 });
 gridObj.appendTo('#Grid');`

| Triggers when context menu item is clicked | **Event:** *contextClick*

 `$("#Grid").ejGrid({

 contextClick: function (args){}
 });` | **Event:** *contextMenuClick*

 `var gridObj =
new ej.grids.Grid({
 contextMenuClick: function (args){}
 });`

| Triggers when context menu opens | **Event:** *contextOpen*

 `$("#Grid").ejGrid({

contextOpen: function (args){}
 });` | **Event:** *contextMenuOpen*

 `var gridObj = new
ej.grids.Grid({
 contextMenuOpen: function (args){}
 });`

Toolbar

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Print | **Property:** *toolbarSettings.toolbarItems*

 `$("#Grid").ejGrid({
 showToolbar: true,
 toolbarItems:
[ej.Grid.ToolbarItems.PrintGrid]
 });` | **Property:** *toolbar*

 `var gridObj = new
ej.grids.Grid({
 toolbar: ['Print']
 });
 gridObj.appendTo('#Grid');`

| Add | **Property:** *toolbarSettings.toolbarItems*

 `$("#Grid").ejGrid({
 showToolbar: true,
 toolbarItems:`

[ej.Grid.ToolBarItems.Add]
}); | **Property:** *toolbar*

 var gridObj = new ej.grids.Grid({
 toolbar: ['Add']
});
gridObj.appendTo('#Grid');

| Edit | **Property:** *toolbarSettings.toolbarItems*

 \$("#Grid").ejGrid({
 showToolbar: true,
 toolbarItems: [ej.Grid.ToolBarItems.Edit]
}); | **Property:** *toolbar*

 var gridObj = new ej.grids.Grid({
 toolbar: ['Edit']
});
gridObj.appendTo('#Grid');

| Delete | **Property:** *toolbarSettings.toolbarItems*

 \$("#Grid").ejGrid({
 showToolbar: true,
 toolbarItems: [ej.Grid.ToolBarItems.Delete]
}); | **Property:** *toolbar*

 var gridObj = new ej.grids.Grid({
 toolbar: ['Delete']
});
gridObj.appendTo('#Grid');

| Update | **Property:** *toolbarSettings.toolbarItems*

 \$("#Grid").ejGrid({
 showToolbar: true,
 toolbarItems: [ej.Grid.ToolBarItems.Update]
}); | **Property:** *toolbar*

 var gridObj = new ej.grids.Grid({
 toolbar: ['Update']
});
gridObj.appendTo('#Grid');

| Cancel | **Property:** *toolbarSettings.toolbarItems*

 \$("#Grid").ejGrid({
 showToolbar: true,
 toolbarItems: [ej.Grid.ToolBarItems.Cancel]
}); | **Property:** *toolbar*

 var gridObj = new ej.grids.Grid({
 toolbar: ['Cancel']
});
gridObj.appendTo('#Grid');

| ExcelExport | **Property:** *toolbarSettings.toolbarItems*

 \$("#Grid").ejGrid({
 showToolbar: true,
 toolbarItems: [ej.Grid.ToolBarItems.ExcelExport]
}); | **Property:** *toolbar*

 var gridObj = new ej.grids.Grid({
 toolbar: ['ExcelExport']
});
gridObj.appendTo('#Grid');

| WordExport | **Property:** *toolbarSettings.toolbarItems*

 \$("#Grid").ejGrid({
 showToolbar: true,
 toolbarItems: [ej.Grid.ToolBarItems.WordExport]
}); | Not Applicable

| PdfExport | **Property:** *toolbarSettings.toolbarItems*

 \$("#Grid").ejGrid({
 showToolbar: true,
 toolbarItems: [ej.Grid.ToolBarItems.PdfExport]
}); | **Property:** *toolbar*

 var gridObj = new ej.grids.Grid({
 toolbar: ['PdfExport']
});
gridObj.appendTo('#Grid');

| Refresh Toolbar | **Method:** *refreshToolbar()*

 var gridObj = \$("#Grid").data("ejGrid");
gridObj.refreshToolbar(); | Not Applicable

| Triggers when toolbar item is clicked | **Event:** *toolbarClick*

 \$("#Grid").ejGrid({
 toolbarClick: function (args){}
 }); | **Event:** *toolbarClick*

 var gridObj = new ej.grids.Grid({
 toolbarClick: function (args){}
 });

GridLines

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| ----- | ----- | ----- |

| Default | **Property:** *gridLines*

 \$("#Grid").ejGrid({
 gridLines: ej.Grid.GridLines.Both
 }); | **Property:** *gridLines*

 var gridObj = new ej.grids.Grid({
 gridLines: 'Default'
 });
gridObj.appendTo('#Grid');

Templates

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Detail Template | **Property:** *detailsTemplate*

 `$("#Grid").ejGrid({
#160; detailsTemplate : "#detailsTemplate"
 });` | **Property:** *detailTemplate*

 `var gridObj = new ej.grids.Grid({
#160;detailTemplate: '#detailtemplate',
});
gridObj.appendTo('#Grid');` |

| Row Template | **Property:** *rowTemplate*

 `$("#Grid").ejGrid({
#160; rowTemplate : "#rowTemplate"
 });` | **Property:** *rowTemplate*

 `var gridObj = new ej.grids.Grid({
#160;rowTemplate: '#rowtemplate',
});
gridObj.appendTo('#Grid');` |

| Refresh Template | **Method:** *refreshTemplate()*

 `var gridObj = $("#Grid").data("ejGrid");
gridObj.refreshTemplate();` | Not Applicable

| Triggers when detail template row is clicked to collapse | **Event:** *detailsCollapse*

 `$("#Grid").ejGrid({
#160; detailsCollapse: function (args){}
 });` | Not Applicable

| Triggers when detail template row is initialized | **Event:** *detailsDataBound*

 `$("#Grid").ejGrid({
#160; detailsDataBound: function (args){}
 });` | Not Applicable

| Triggers when detail template
row is clicked to expand | **Event:** *detailsExpand*

 `$("#Grid").ejGrid({
#160; detailsExpand: function (args){}
 });` | **Event:** *detailDataBound*

 `var gridObj = new ej.grids.Grid({
#160; detailDataBound: function (args){}
 });`

| Triggers when refresh the template column elements in the Grid | **Event:** *templateRefresh*

 `$("#Grid").ejGrid({
#160; templateRefresh: function (args){}
 });` | Not Applicable

Row/Column Drag and Drop

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *allowRowDragAndDrop*

 `$("#Grid").ejGrid({
#160; allowRowDragAndDrop: true
 });` | **Property:** *allowRowDragAndDrop*

 `var gridObj = new ej.grids.Grid({
#160;allowRowDragAndDrop: true,
#160;rowDropSettings: { targetID: 'DestGrid' }
});
gridObj.appendTo('#Grid');` |

| Triggers when the row is
being dragged | **Event:** *rowDrag*

 `$("#Grid").ejGrid({
#160; rowDrag: function (args){}
 });` | **Event:** *rowDrag*

 `var gridObj = new ej.grids.Grid({
#160; rowDrag: function (args){}
 });`

| Triggers when the row drag begins | **Event:** *rowDragStart*

 `$("#Grid").ejGrid({
#160; rowDragStart: function (args){}
 });` | **Event:** *rowDragStart*

 `var gridObj = new ej.grids.Grid({
#160; rowDragStart: function (args){}
 });`

| Triggers when the row is dropped | **Event:** *rowDrop*

 `$("#Grid").ejGrid({
#160; rowDrop: function (args){}
 });` | **Event:** *rowDrop*

 `var gridObj = new ej.grids.Grid({
#160; rowDrop: function (args){}
 });`

| Triggers before the row is being dropped | **Event:** *beforeRowDrop*

 `$("#Grid").ejGrid({

 beforeRowDrop: function (args){}
 });` | Not Applicable

| Triggers when the
column is being dragged | **Event:** *columnDrag*

 `$("#Grid").ejGrid({

 columnDrag: function (args){}
 });` | Not Applicable

| Triggers when the
column drag begins | **Event:** *columnDragStart*

 `$("#Grid").ejGrid({

 columnDragStart: function (args){}
 });` | Not Applicable

| Triggers when the
column is dropped | **Event:** *columnDrop*

 `$("#Grid").ejGrid({

 columnDrop: function (args){}
 });` | Not Applicable

Frozen Rows and Columns

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *scrollSettings.frozenRows*

 `$("#Grid").ejGrid({

allowScrolling: true,
 scrollSettings:{frozenRows:2,
frozenColumns: 2 }
 });` |
Property: *frozenColumns*

`var gridObj = new ej.grids.Grid({
 frozenColumns : 2,

 frozenRows:2
});
gridObj.appendTo('#Grid');` |

| isFrozen | **Property:** *columns.isFrozen*

 `$("#Grid").ejGrid({
 columns : [{ field:
"ShipCity", isFrozen:true}
 });` | **Property:** *columns.isFrozen*

`var gridObj = new
ej.grids.Grid({
 columns : [{ field: "ShipCity", isFrozen: true}

});
gridObj.appendTo('#Grid');` |

ForeignKey

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *columns.foreignKeyValue*

 `$("#Grid").ejGrid({
 columns : [{
field: "EmployeeID", foreignKeyField: "EmployeeID", foreignKeyValue: "FirstName",
dataSource: window.employeeView, headerText: "First Name"}
 });` | **Property:**
columns.foreignKeyValue

`var gridObj = new ej.grids.Grid({
 columns : [{ field:
'EmployeeID', headerText: 'Employee Name', width: 150, foreignKeyValue: 'FirstName',
dataSource: employeeData}
});
gridObj.appendTo('#Grid');` |

Auto Wrap

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *allowTextWrap*

 `$("#Grid").ejGrid({
 allowTextWrap :
true
 });` | **Property:** *allowTextWrap*

`var gridObj = new ej.grids.Grid({

 allowTextWrap: true,
});
gridObj.appendTo('#Grid');` |

| Both | **Property:** *allowTextWrap*

 `$("#Grid").ejGrid({
 allowTextWrap :
true
 textWrapSettings: {wrapMode: "both"},
 });` | **Property:** *allowTextWrap*

`var gridObj = new ej.grids.Grid({
 allowTextWrap: true,

 textWrapSettings: { wrapMode: 'Both' }
});
gridObj.appendTo('#Grid');` |

| Header | **Property:** *allowTextWrap*

 `$("#Grid").ejGrid({
 allowTextWrap : true
 textWrapSettings: {wrapMode: "header"},
 });` | **Property:** *allowTextWrap*

 `var gridObj = new ej.grids.Grid({
 allowTextWrap: true,
 textWrapSettings: { wrapMode: 'Header' }
});
gridObj.appendTo('#Grid');` |

| Content | **Property:** *allowTextWrap*

 `$("#Grid").ejGrid({
 allowTextWrap : true
 textWrapSettings: {wrapMode: "content"},
 });` | **Property:** *allowTextWrap*

 `var gridObj = new ej.grids.Grid({
 allowTextWrap: true,
 textWrapSettings: { wrapMode: 'Content' }
});
gridObj.appendTo('#Grid');` |

Responsive

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *isResponsive*

 `$("#Grid").ejGrid({
 isResponsive: true,enableResponsiveRow: true
});` | Not Applicable

State Persistence

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *enablepersistence*

 `$("#Grid").ejGrid({
 enablepersistence: true
 });` | **Property:** *enablepersistence*

 `var gridObj = new ej.grids.Grid({
 enablepersistence: true
});
gridObj.appendTo('#Grid');` |

Right to Left - RTL

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *enableRTL*

 `$("#Grid").ejGrid({
 enableRTL: true
 });` | **Property:** *enableRtl*

 `var gridObj = new ej.grids.Grid({
 enableRtl: true
});
gridObj.appendTo('#Grid');` |

ToolTip

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *clipMode*

 `$("#Grid").ejGrid({
 columns : [{ field: "ShipCity", clipMode: ej.Grid.ClipMode.EllipsisWithTooltip}
 { field: "ShipCountry", clipMode: ej.Grid.ClipMode.Ellipsis}
 { field: "ShipName", clipMode: ej.Grid.ClipMode.Clip}]
 });` | **Property:** *clipMode*

 `var gridObj = new ej.grids.Grid({
 columns : [{ field: "ShipCity", clipMode: 'EllipsisWithTooltip'},
 { field: "ShipCountry", clipMode: 'Ellipsis'},
 { field: "ShipName", clipMode: 'Clip'}
});
gridObj.appendTo('#Grid');` |

Aggregate/Summary

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Footer Aggregate | **Property:** *showSummary*

 `$("#Grid").ejGrid({
 showSummary: true
 summaryRows: [{
 title: "Sum",
 summaryColumns: [{
 summaryType: ej.Grid.SummaryType.Sum,
 displayColumn: "Freight",
 dataMember: "Freight",
 format: "{0:C2}"}]})`; | **Property:** *aggregates*

 `var gridObj = new ej.grids.Grid({
 aggregates: [{
 columns: [{
 type: 'Sum',
 field: 'Freight',
 footerTemplate: 'Sum: ${Sum}'}]})`;
);
gridObj.appendTo("#Grid"); |

| Caption Aggregate | **Property:** *showSummary*

 `$("#Grid").ejGrid({
 showSummary: true
 summaryRows: [{
 showCaptionSummary: true,
 title: "Sum",
 summaryColumns: [{
 summaryType: ej.Grid.SummaryType.Sum,
 displayColumn: "Freight",
 dataMember: "Freight",
 format: "{0:C2}"}]})`; | **Property:** *aggregates*

 `var gridObj = new ej.grids.Grid({
 aggregates: [{
 columns: [{
 type: 'Sum',
 field: 'Freight',
 groupCaptionTemplate: 'Max: ${Max}'}]})`;
);
gridObj.appendTo("#Grid"); |

| Get Summary values | **Method:** *getSummaryValues(summaryCol, summaryData)*

 `var gridObj = $("#Grid").data("ejGrid");
 gridObj.getSummaryValues(summaryCol, window.gridData);` | Not Applicable

Grid Export

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Adds a grid model property which is
to be ignored on exporting grid | **Method:** *addIgnoreOnExport(propertyNames)*

 `var gridObj = $("#Grid").data("ejGrid");
 gridObj.addIgnoreOnExport("filterSettings");` | Not Applicable |

Columns

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Add or Remove Columns | **Method:** *columns(columnDetails, [action])()*

 `var gridObj = $("#Grid").data("ejGrid");
 gridObj.columns("OrderID", "remove");
 gridObj.columns("CustomerID", "add");` | **Property:** *columns*

 Grid is initially rendered with OrderID and CustomerId columns. Then if you want to add ShipAddress column, you have to reset the value for column property as `gridObj.columns = [{field:"OrderID"}, {field:"CustomerId"}, {field:"ShipAddress"}]`; Then to remove the CustomerId column, reset the column property as, `gridObj.columns = [{field:"OrderID"}, {field:"ShipAddress"}]`;

| Get Column By Field | **Method:** *getColumnByField(fieldName)*

 `var gridObj = $("#Grid").data("ejGrid");
 gridObj.getColumnByField("OrderID");` | **Method:** *getColumnByField(fieldName)*

 `var gridObj =`

```
document.getElementById('Grid')<br>.ej2_instances[0]
<br>gridObj.getColumnByField("OrderID")
```

| Get Column By HeaderText | **Method:** *getColumnByHeaderText(headerText)*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.getColumnByHeaderText("Order ID"); | Column object for a corresponding headerText can able to get by iterating the gridObj.columns with the headerText

| Get Column By Index | **Method:** *getColumnByIndex(columnIndex)*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.getColumnByIndex(1); | **Method:** *getColumnByIndex(columnIndex)*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.getColumnByIndex(1)

| Get Column Fieldnames | **Method:** *getColumnFieldNames()*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.getColumnFieldNames(); | **Method:** *getColumnFieldNames()*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.getColumnFieldNames()

| Get Column Index By Field | **Method:** *getColumnIndexByField(fieldName)*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.getColumnIndexByField("OrderID"); | **Method:** *getColumnIndexByField(fieldName)*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.getColumnIndexByField("OrderID");

| Get Column Index By headerText | **Method:** *getColumnIndexByHeaderText(headerText, [field])*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.getColumnIndexByHeaderText("Order ID"); | The Column Index for a corresponding headerText can be get by iterating the gridObj.columns with the headerText

| Set Width to columns | **Method:** *setWidthToColumns()*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.setWidthToColumns(); | **Method:** *widthService.setWidthToColumns()*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.widthService.setWidthToColumns()

| Get HeaderText By FieldName | **Method:** *getHeaderTextByFieldName()*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.getHeaderTextByFieldName("OrderID"); | Not Applicable

| Get Hidden Columnname | **Method:** *getHiddenColumnNames()*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.getHiddenColumnNames(); | Not Applicable

| Get Visible Columns/Names | **Method:** *getVisibleColumnNames()*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.getVisibleColumnNames(); | **Method:** *getVisibleColumns()*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.getVisibleColumns();

| Select Columns | **Method:** *selectColumns(fromIndex)*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.selectColumns(1,4); | Not Applicable

| Select Specific Columns based on Index | **Method:** *selectColumns(columnIndex,[toIndex])*

 var gridObj = \$("#Grid").data("ejGrid");
 gridObj.selectColumns(1,4); | Not Applicable

Row

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| EnableHover | **Property:** *enableRowHover*

 `$("#Grid").ejGrid({
 enableRowHover : false
 });`| **Property:** *enableHover*

 `var gridObj = new ej.grids.Grid({
 enableHover: false,
});
gridObj.appendTo('#Grid');` |

| Get Row Height | **Method:** *getRowHeight()*

 `var gridObj = $("#Grid").data("ejGrid");
gridObj.getRowHeight();`| **Method:** *getRowHeight()*

 `var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.getRowHeight();`

| Refresh Row Height | **Method:** *rowHeightRefresh()*

 `var gridObj = $("#Grid").data("ejGrid");
gridObj.rowHeightRefresh();`| Not Applicable

| Get index by Row Element | **Method:** *getIndexByRow(\$tr)*

 `var gridObj = $("#Grid").data("ejGrid");
gridObj.getIndexByRow($(".gridcontent tr").first());`| Not Applicable

| Get Row by its Index | **Method:** *getRowByIndex(from, to)*

 `var gridObj = $("#Grid").data("ejGrid");
gridObj.getRowByIndex(3, 6);`| **Method:** *getRowByIndex()*

 `var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.getRowByIndex(1);`

| Get rendered rows | **Method:** *getRows()*

 `var gridObj = $("#Grid").data("ejGrid");
gridObj.getRows();`| **Method:** *getRows()*

 `var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.getRows();`

| Triggers while hover the grid row | **Event:** *rowHover*

 `$("#Grid").ejGrid({
 rowHover: function (args){}
 });`| Not Applicable

Show/Hide Columns

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Hide Columns by using method | **Method:** *hideColumns(headerText)*

 `var gridObj = $("#Grid").data("ejGrid");
gridObj.hideColumns("Order ID");`| **Method:** *hideColumns(headerText)*

 `var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.hideColumns("Order ID");`

| Show Columns by using method | **Method:** *showColumns(headerText)*

 `var gridObj = $("#Grid").data("ejGrid");
gridObj.showColumns("Order ID");`| **Method:** *showColumns(headerText)*

 `var gridObj = document.getElementById('Grid')
.ej2_instances[0]
gridObj.showColumns("Order ID");`

Column Chooser

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Default | **Property:** *showColumnChooser*

 `$("#Grid").ejGrid({
 showColumnChooser : true
 columns: [
 { field: 'CustomerID', showInColumnChooser: false },
 { field: 'Name', showInColumnChooser: true },
]});` | **Property:** *showColumnChooser*

 `var gridObj = new ej.grids.Grid({
 showColumnChooser: true,
 toolbar: ['ColumnChooser'],
 columns: [{
 { field: 'CustomerID', showInColumnChooser: false },
 { field: 'Name', showInColumnChooser: true },
 }]);
 gridObj.appendTo('#Grid');` |

Header

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Refresh Header | **Method:** *refreshHeader()*

 `var gridObj = $("#Grid").data("ejGrid");
 gridObj.refreshHeader();` | **Method:** *refreshHeader()*

 `var gridObj = document.getElementById('Grid')
 .ej2_instances[0]
 gridObj.refreshHeader();`

| Triggers every time a request is made to access particular header cell information, element and data. |

Event: *mergeHeaderCellInfo*

 `$("#Grid").ejGrid({
 mergeHeaderCellInfo: function (args){}
 });` | Not Applicable

| Triggers every time a request is made to access particular cell information, element and data | **Event:** *mergeCellInfo*

 `$("#Grid").ejGrid({
 mergeCellInfo: function (args){}
 });` | Not Applicable

DataSource

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| DataSource | **Method:** *dataSource(datasource,[templateRefresh])*

 `var gridObj = $("#Grid").data("ejGrid");
 gridObj.dataSource(newdataSource);` | **Property:** *dataSource*

 `var gridObj = document.getElementById('Grid')
 .ej2_instances[0]
 gridObj.dataSource = newdataSource`

Print

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Print the grid | **Method:** *print()*

 `var gridObj = $("#Grid").data("ejGrid");
 gridObj.print();` | **Method:** *print()*

 `var gridObj = document.getElementById('Grid')
 .ej2_instances[0]
 gridObj.print();`

| Triggers before printing the grid | **Event:** *beforePrint*

 `$("#Grid").ejGrid({
 `

`beforePrint: function (args){}
 });` | **Event:** *beforePrint*

 `var gridObj = new ej.grids.Grid({
 beforePrint: function (args){}
 });`

Scrolling

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Get ScrollObject | **Method:** *getScrollObject()*

 `var gridObj = $("#Grid").data("ejGrid");`

`gridObj.getScrollObject();` | **Property:** *grid.scrollModule*

 `var gridObj =`
`document.getElementById('Grid')`
 `.ej2_instances[0]`
 `var scrollObj =`
`gridObj.scrollModule;`

PrimaryKey

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Set PrimaryKey Column | **Property:** *columns.isPrimaryKey*

 `$("#Grid").ejGrid({`
 `columns:`
`[`
 `{ field: "OrderID", isPrimaryKey: true}]`
 `});` | **Property:** *columns.isPrimaryKey*

 `var`
`gridObj = new ej.grids.Grid({`
 `columns: [`
 `{ field: "OrderID", isPrimaryKey: true}]`
 `});`

| Get the PrimaryKey fieldnames | **Method:** *getPrimaryKeyFieldNames()*

 `var gridObj =`
`$("#Grid").data("ejGrid");`
 `gridObj.getPrimaryKeyFieldNames();` | **Method:**
`getPrimaryKeyFieldNames()`

 `var gridObj =`
`document.getElementById('Grid')`
 `.ej2_instances[0]`

`gridObj.getPrimaryKeyFieldNames();`

Grid

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Get the Browser Details | **Method:** *getBrowserDetails()*

 `var gridObj =`
`$("#Grid").data("ejGrid");`
 `gridObj.getBrowserDetails();` | In Essential JS 2, it can be

 achieved by using `Browser` class of `ej2-base`

| Set dimension for the grid | **Method:** *setDimension(height, width)*

 `var gridObj =`
`$("#Grid").data("ejGrid");`
 `gridObj.setDimension(300,400);` | Not Applicable

| set maximum width for mobile | **Method:** *setPhoneModeMaxWidth(value)*

 `var gridObj =`
`$("#Grid").data("ejGrid");`
 `gridObj.setPhoneModeMaxWidth(500);` | Not Applicable

| Render the grid | **Method:** *render()*

 `var gridObj = $("#Grid").data("ejGrid");`

`gridObj.render();` | **Method:** *render()*

 `var gridObj =`
`document.getElementById('Grid')`
 `.ej2_instances[0]`
 `gridObj.render();`

| Reset the model collections like `pageSettings`, `groupSettings`, `filterSettings`, `sortSettings` and
`summaryRows`. | **Method:** *resetModelCollections()*

 `var gridObj = $("#Grid").data("ejGrid");`

 `gridObj.resetModelCollections();` | Not Applicable

| Destroy the grid | **Method:** *destroy()*

 `var gridObj = $("#Grid").data("ejGrid");`

`gridObj.destroy();` | **Method:** *destroy()*

 `var gridObj =`
`document.getElementById('Grid')`
 `.ej2_instances[0]`
 `gridObj.destroy();`

| Get Content Element | **Method:** *getContent()*

 `var gridObj = $("#Grid").data("ejGrid");`

`gridObj.getContent();` | **Method:** *getContent()*

 `var gridObj =`
`document.getElementById('Grid')`
 `.ej2_instances[0]`
 `gridObj.getContent();`

| Get Content Table | **Method:** *getContentTable()*

 var gridObj = \$("#Grid").data("ejGrid");

 gridObj.getContentTable(); | **Method:** *getContentTable()*

 var gridObj =
document.getElementById('Grid')
.ej2_instances[0]
 gridObj.getContentTable();

| Get Current View Data | **Method:** *getCurrentViewData()*

 var gridObj =
\$("#Grid").data("ejGrid");
 gridObj.getCurrentViewData(); | **Method:** *getCurrentViewRecords()*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]

gridObj.getCurrentViewRecords();

| Get Data Row | **Method:** *getDataByIndex(rowIndex)*

 var gridObj =
\$("#Grid").data("ejGrid");
 gridObj.getDataByIndex(0); | **Method:** *getDataRows()*

 var
gridObj = document.getElementById('Grid')
.ej2_instances[0]
 gridObj.getDataRows();

| Get Fieldname by using the headertext | **Method:** *getFieldNameByHeaderText(headerText)*

var gridObj = \$("#Grid").data("ejGrid");
 gridObj.getFieldNameByHeaderText("Order ID"); |
Not Applicable

| Get FooterContent | **Method:** *getFooterContent()*

 var gridObj = \$("#Grid").data("ejGrid");

 gridObj.getFooterContent(); | **Method:** *getFooterContent()*

 var gridObj =
document.getElementById('Grid')
.ej2_instances[0]
 gridObj.getFooterContent();

| Get FooterContent Table | **Method:** *getFooterTable()*

 var gridObj =
\$("#Grid").data("ejGrid");
 gridObj.getFooterTable(); | **Method:** *getFooterContentTable()*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]

gridObj.getFooterContentTable();

| Get HeaderComponent | **Method:** *getHeaderContent()*

 var gridObj =
\$("#Grid").data("ejGrid");
 gridObj.getHeaderContent(); | **Method:** *getHeaderContent()*

 var gridObj = document.getElementById('Grid')
.ej2_instances[0]

gridObj.getHeaderContent();

| Get HeaderComponent Table | **Method:** *getHeaderTable()*

 var gridObj =
\$("#Grid").data("ejGrid");
 gridObj.getHeaderTable(); | **Method:** *getHeaderTable()*

 var
gridObj = document.getElementById('Grid')
.ej2_instances[0]

gridObj.getHeaderTable();

| Refresh Content | **Method:** *refreshContent()*

 var gridObj = \$("#Grid").data("ejGrid");

gridObj.refreshContent(); | **Method:**
 contentModule.refreshContentRows()

 var gridObj
= document.getElementById('Grid')
.ej2_instances[0]

gridObj.
contentModule.refreshContentRows();

| Refresh Data | **Method:** *refreshData()*

 var gridObj = \$("#Grid").data("ejGrid");

gridObj.refreshData(); | Not Applicable

| Triggers every time a request is
made to access particular cell
information, element and data |
Event: *queryCellInfo*

 \$("#Grid").ejGrid({
 queryCellInfo: function (args){}

}); | **Event:** *queryCellInfo*

 var gridObj = new ej.grids.Grid({
 queryCellInfo:
function (args){}
 });

| Triggers every time a request is
made to access row information,
element and data | **Event:**
rowDataBound

 \$("#Grid").ejGrid({
 rowDataBound: function (args){}
 }); |

Event: *rowDataBound*

 `var gridObj = new ej.grids.Grid({
 rowDataBound: function (args){
 }
 });`

| Triggers for every grid
action before it get started | **Event:** *actionBegin*

 `$("#Grid").ejGrid({
 actionBegin: function (args){
 }
 });` | **Event:** *actionBegin*

 `var gridObj = new ej.grids.Grid({
 actionBegin: function (args){
 }
 });`

| Triggers for every grid
action success event | **Event:** *actionComplete*

 `$("#Grid").ejGrid({
 actionComplete: function (args){
 }
 });` | **Event:** *actionComplete*

 `var gridObj = new ej.grids.Grid({
 actionComplete: function (args){
 }
 });`

| Triggers for every grid
server failure event | **Event:** *actionFailure*

 `$("#Grid").ejGrid({
 actionFailure: function (args){
 }
 });` | **Event:** *actionFailure*

 `var gridObj = new ej.grids.Grid({
 actionFailure: function (args){
 }
 });`

| Triggers when the grid is bound
with data during rendering | **Event:** *dataBound*

 `$("#Grid").ejGrid({
 dataBound: function (args){
 }
 });` | **Event:** *dataBound*

 `var gridObj = new ej.grids.Grid({
 dataBound: function (args){
 }
 });`

| Triggers when the grid is
going to destroy | **Event:** *destroy*

 `$("#Grid").ejGrid({
 destroy: function (args){
 }
 });` | **Event:** *destroyed*

 `var gridObj = new ej.grids.Grid({
 destroyed: function (args){
 }
 });`

| Triggers when initial load of the grid | **Event:** *load*

 `$("#Grid").ejGrid({
 load: function (args){
 }
 });` | **Event:** *load*

 `var gridObj = new ej.grids.Grid({
 load: function (args){
 }
 });`

| Triggers when the grid is rendered completely | **Event:** *create*

 `$("#Grid").ejGrid({
 create: function (args){
 }
 });` | **Event:** *created*

 `var gridObj = new ej.grids.Grid({
 created: function (args){
 }
 });`

| Triggers when the record is clicked | **Event:** *recordClick*

 `$("#Grid").ejGrid({
 recordClick: function (args){
 }
 });` | Not Applicable

| Triggers when right clicked on grid element | **Event:** *rightClick*

 `$("#Grid").ejGrid({
 rightClick: function (args){
 }
 });` | Not Applicable

| Triggers when the
record is double clicked | **Event:** *recordDoubleClick*

 `$("#Grid").ejGrid({
 recordDoubleClick: function (args){
 }
 });` | **Event:** *recordDoubleClick*

 `var gridObj = new ej.grids.Grid({
 recordDoubleClick: function (args){
 }
 });`

How To

Enable/disable grid and its actions in EJ2 JavaScript Grid control

You can enable/disable the Grid and its actions by applying/removing corresponding CSS styles.

To enable/disable the grid and its actions, follow the given steps:

Step 1:

Create CSS class with custom style to override the default style of Grid.

,

```
.disablegrid {
pointer-events: none;
opacity: 0.4;
}

.wrapper {
cursor: not-allowed;
}
`
```

Step 2:

Add/Remove the CSS class to the Grid in the click event handler of Button.

```
`ts
document.getElementById('primarybtn').addEventListener('click', () => {
if (grid.element.classList.contains('disablegrid')) {
grid.element.classList.remove('disablegrid');
document.getElementById("GridParent").classList.remove('wrapper');
}
else {
grid.element.classList.add('disablegrid');
document.getElementById("GridParent").classList.add('wrapper');
}
});
`
```

In the below demo, the button click will enable/disable the Grid and its actions.

INDEX.TS

```
import { Grid, Edit, Toolbar, EditEventArgs, EJ2Intance } from
 '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let button: Button = new Button();
button.appendTo('#primarybtn');
let grid: Grid = new Grid({
  dataSource: data,
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  editSettings: { allowAdding: true, allowEditing: true, allowDeleting: true
},
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
```

```

        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    height: 265
});
grid.appendTo('#Grid');
document.getElementById('primarybtn').addEventListener('click', () => {
    if (grid.element.classList.contains('disablegrid')) {
        grid.element.classList.remove('disablegrid');
        document.getElementById("GridParent").classList.remove('wrapper');
    }
    else {
        grid.element.classList.add('disablegrid');
        document.getElementById("GridParent").classList.add('wrapper');
    }
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="primarybtn">Enable/Disable Grid</button>
        <div id="GridParent">
            <div id="Grid"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the edit dialog in EJ2 JavaScript Grid control

You can customize the appearance of the edit dialog in the [actionComplete](#) event based on `requestType` as `beginEdit` or `add`.

In the following example, the dialog's properties like header text, `showCloseIcon`, height have been changed while editing and adding the records.

Also the locale text for the `Save` and `Cancel` buttons has been changed by overriding the default locale strings.

You can refer the Grid [Default text](#) list for more localization.

INDEX.TS

```

import { L10n } from '@syncfusion/ej2-base';
import { DialogModel } from '@syncfusion/ej2-popups';
import { Grid, Edit, Toolbar, DialogEditEventArgs } from '@syncfusion/ej2-
grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
L10n.load({
    'en-US': {
        grid: {
            SaveButton: 'Submit',
            CancelButton: 'Discard'
        }
    }
});
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog' },

```

```

        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
            { field: 'CustomerID', headerText: 'Customer ID', width: 120, },
            { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
        ],
        height: 265,
        actionComplete: (args: DialogEditEventArgs) => {
            if (args.requestType === 'beginEdit' || args.requestType === 'add')
            {
                let dialog: DialogModel = args.dialog;
                dialog.showCloseIcon = false;
                dialog.height = 400;
                // change the header of the dialog
                dialog.header = args.requestType === 'beginEdit' ? 'Edit Record
of ' + args.rowData['CustomerID'] : 'New Customer';
            }
        }
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cascading drop down list with grid editing in EJ2 JavaScript Grid control

You can achieve the Cascading DropDownList with grid Editing by using the Cell Edit Template feature.

In the below demo, Cascading DropDownList is rendered for the **ShipCountry** and **ShipState** column.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Query } from '@syncfusion/ej2-data';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let countryElem: HTMLElement;
let countryObj: DropDownList;
let stateElem: HTMLElement;
let stateObj: DropDownList;
let country: { [key: string]: Object }[] = [
    { countryName: 'United States', countryId: '1' },
    { countryName: 'Australia', countryId: '2' }
];
let state: { [key: string]: Object }[] = [
    { stateName: 'New York', countryId: '1', stateId: '101' },
    { stateName: 'Virginia ', countryId: '1', stateId: '102' },
    { stateName: 'Washington', countryId: '1', stateId: '103' },
    { stateName: 'Queensland', countryId: '2', stateId: '104' },
    { stateName: 'Tasmania ', countryId: '2', stateId: '105' },
    { stateName: 'Victoria', countryId: '2', stateId: '106' }
];
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
    columns: [

```



```

        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true } },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120,
validationRules: { required: true, minLength: 3 } },
        {
            field: 'ShipCountry', headerText: 'Ship Country', width: 120,
edit: {
                create: () => {
                    countryElem = document.createElement('input');
                    return countryElem;
                },
                read: () => {
                    return countryObj.text;
                },
                destroy: () => {
                    countryObj.destroy();
                },
                write: () => {
                    countryObj = new DropDownList({
                        dataSource: country,
                        fields: { value: 'countryId', text: 'countryName' },
                        change: () => {
                            stateObj.enabled = true;
                            let tempQuery: Query = new
Query().where('countryId', 'equal', countryObj.value);
                            stateObj.query = tempQuery;
                            stateObj.text = null;
                            stateObj.dataBind();
                        },
                        placeholder: 'Select a country',
                        floatLabelType: 'Never'
                    });
                    countryObj.appendTo(countryElem);
                }
            },
        },
        {
            field: 'ShipState', headerText: 'Ship State', width: 150, edit:
{
                create: () => {
                    stateElem = document.createElement('input');
                    return stateElem;
                },
                read: () => {
                    return stateObj.text;
                },
                destroy: () => {
                    stateObj.destroy();
                },
                write: () => {
                    stateObj = new DropDownList({
                        dataSource: state,
                        fields: { value: 'stateId', text: 'stateName' },
                        enabled: false,
                        placeholder: 'Select a state',
                        floatLabelType: 'Never'
                    });
                }
            }
        }
    ],
    edit: {
        create: () => {
            stateElem = document.createElement('input');
            return stateElem;
        },
        read: () => {
            return stateObj.text;
        },
        destroy: () => {
            stateObj.destroy();
        },
        write: () => {
            stateObj = new DropDownList({
                dataSource: state,
                fields: { value: 'stateId', text: 'stateName' },
                enabled: false,
                placeholder: 'Select a state',
                floatLabelType: 'Never'
            });
        }
    }
}

```

```

        stateObj.appendTo(stateElem);
    }
}
},
height: 265
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
    }
  </style>

```

```

        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Perform grid actions by keyboard short cut keys in EJ2 JavaScript Grid control

Using keyboard shortcuts, Grid performs navigation and actions.

In addition, You can also perform grid actions with custom keyboard shortcuts. This operation has to be achieved outside of the grid with the help of `keydown` event.

The following example demonstrates on Adding a new row when `Enter` key is pressed in the grid.

INDEX.TS

```

import { Grid, Edit, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true, validationRules: { required: true } },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 265
});
grid.appendTo('#Grid');
grid.element.addEventListener("keydown", keyDownHandler);
function keyDownHandler(e: any): void {
    if(e.keyCode === 13) {
        let gridIns: any = document.getElementById("Grid").ej2_instances[0];

```

```

        gridIns.addRecord();
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
        .empImage {
            margin: 6px 16px;
            float: left;
            width: 50px;
            height: 50px;
        }
    </style>

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting grid in cordova application in EJ2 JavaScript Grid control

Cordova application does not support direct file download. So we have to use the Blob stream to export the Grid.

You can use corresponding exporting methods and exportComplete events to get the Blob stream.

INDEX.TS

```

import { EmitType } from '@syncfusion/ej2-base'
import { Grid, Page, Toolbar, ExcelExport, ExcelExportCompleteArgs,
PdfExport, PdfExportCompleteArgs } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar, ExcelExport, PdfExport);
/**
 * Exporting Blob data
 */
let excelExpComplete: EmitType<ExcelExportCompleteArgs> = (args:
ExcelExportCompleteArgs) => {
    //This event will be triggered when excel exporting.
    args.promise.then((e: { blobData: Blob }) => {
        //In this `then` function, we can get blob data through the
arguments after promise resolved.
        exportBlob(e.blobData);
    });
};
let pdfExpComplete: EmitType<PdfExportCompleteArgs> = (args:
PdfExportCompleteArgs) => {
    //This event will be triggered when pdf exporting.
    args.promise.then((e: { blobData: Blob }) => {
        //In this `then` function, we can get blob data through the
arguments after promise resolved.
        exportBlob(e.blobData);
    });
};
let exportBlob: Function = (blob: Blob) => {
    let a: HTMLAnchorElement = document.createElement('a');

```

```

        document.body.appendChild(a);
        a.style.display = 'none';
        let url: string = window.URL.createObjectURL(blob);
        a.href = url;
        a.download = 'Export';
        a.click();
        window.URL.revokeObjectURL(url);
        document.body.removeChild(a);
    }
    let grid: Grid = new Grid(
    {
        dataSource: data,
        allowExcelExport: true,
        allowPdfExport: true,
        excelExportComplete: excelExpComplete,
        pdfExportComplete: pdfExpComplete,
        toolbar: ['PdfExport', 'ExcelExport'],
        columns: [
            {
                field: 'OrderID', headerText: 'Order ID', textAlign:
'Right', width: 90
            },
            {
                field: 'CustomerID', headerText: 'Customer ID', width: 90
            },
            { field: 'ShipCountry', headerText: 'ShipCountry', width: 90 },
        ]
    });
    grid.appendTo('#Grid');
    grid.toolbarClick = (args: Object) => {
        if (args['item'].id === 'Grid_pdfexport') {
            grid.pdfExport(null, null, null, true);
        }
        if (args['item'].id === 'Grid_excelexport') {
            grid.excelExport(null, null, null, true);
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize pager drop down in EJ2 JavaScript Grid control

To customize the default values of pager drop-down, you need to define the [pageSizes](#) as array of strings.

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    pageSettings: { pageSizes: ["5", "10", "All"] },
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },

```

```

        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```



```

        ele.style.visibility = "visible";
    }
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hide the expand/collapse icon in parent row in EJ2 JavaScript Grid control

By default, the expand/collapse icon will be visible even if the child grid is empty.

You can use [rowDataBound](#) event to hide the icon when there is no record in child grid.

To hide the expand/collapse icon in parent row with no record in child grid, follow the given steps:

Step 1:

Create CSS class with custom style to override the default style of Grid.

```

,

.e-row[aria-selected="true"] .e-customizedExpandcell {
background-color: #e0e0e0;
}
.e-grid.e-gridhover tr[role='row']:hover {
background-color: #eee;
}
,

```

Step 2:

Add the CSS class to the Grid in the [rowDataBound](#) event handler of Grid.

```

`ts
function rowDataBound(args:any):void{
var filter = args.data.EmployeeID;
var data = new DataManager(this.childGrid.dataSource).executeLocal(new
Query().where("EmployeeID", "equal", parseInt(filter), true));
if(data.length == 0) {
//here hide which parent row has no child records
args.row.querySelector('td').innerHTML=" ";
args.row.querySelector('td').className = "e-customizedExpandcell";
}
}
,

```

In the below demo, the expand/collapse icon in the row with **EmployeeID** as **1** is hidden as it does not have record in child Grid.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { Grid, Page, Selection, DetailRow } from '@syncfusion/ej2-grids';
import { DataManager, Query } from '@syncfusion/ej2-data';
import { employeeData } from './employeeData.ts';
Grid.Inject(Page, Selection, DetailRow);
let dataManger: Object = [{Order:100, ShipName:'Berlin', EmployeeID:2},
    {Order:101, ShipName:'Capte', EmployeeID:3},
    {Order:102, ShipName:'Marlon', EmployeeID:4},
    {Order:103, ShipName:'Black pearl', EmployeeID:5},
    {Order:104, ShipName:'Pearl', EmployeeID:6},
    {Order:105, ShipName:'Noth bay', EmployeeID:7},
    {Order:106, ShipName:'baratna', EmployeeID:8},
    {Order:107, ShipName:'Charge', EmployeeID:9},
];
let grid: Grid = new Grid({
    dataSource: employeeData,
    allowSorting: true,
    rowDataBound:rowDataBound,
    columns: [
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 125 },
        { field: 'FirstName', headerText: 'Name', width: 125 },
        { field: 'Title', headerText: 'Title', width: 180 },
        { field: 'City', headerText: 'City', width: 110 },
        { field: 'Country', headerText: 'Country', width: 110 }
    ],
    childGrid: {
        dataSource: dataManger,
        queryString: 'EmployeeID',
        allowPaging: true,
        columns: [
            { field: 'Order', headerText: 'Order ID', textAlign:
'Right', width: 120 },
            { field: 'EmployeeID', headerText: 'Employee ID', width: 120
},
            { field: 'ShipName', headerText: 'Ship Name', width: 150 }
        ],
    },
});
grid.appendTo('#Grid');
function rowDataBound(args:any):void{
    var filter = args.data.EmployeeID;
    var data = new
DataManager(this.childGrid.dataSource).executeLocal(new
Query().where("EmployeeID", "equal", parseInt(filter), true));
    if(data.length == 0) {
        //here hide which parent row has no child records
        args.row.querySelector('td').innerHTML=" ";
        args.row.querySelector('td').className = "e-
customizedExpandcell";
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Select rows in grid based on certain condition in EJ2 JavaScript Grid control

You can select the specific row in the grid based on a certain condition by using the [selectRows](#) method in the [dataBound](#) event of Grid.

In the below demo, we have selected the grid rows only when **EmployeeID** column value greater than 3.

INDEX.TS

```

import { Grid, Selection, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Selection, Page);
let selIndex: any = [];
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    selectionSettings: {type: "Multiple"},
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
        { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 150 },
        { field: 'ShipCity', headerText: 'Ship City', width: 150 },
        { field: 'ShipName', headerText: 'Ship Name', width: 150 }
    ],
    rowDataBound: function(args) {
        if (args.data['EmployeeID'] > 3) {
            selIndex.push(parseInt(args.row.getAttribute('aria-rowindex')));
        }
    },
    dataBound: function(args) {
        if (selIndex.length) {
            this.selectRows(selIndex);
            selIndex = [];
        }
    }
});
grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Grid"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Collapse grouped rows at initial render in EJ2 JavaScript Grid control

You can collapse all the grouped rows at initial rendering by using `dataBound` event with `collapseAll` method of the grid.

In the below demo, all the grouped rows are collapsed at initial rendering.

INDEX.TS

```
import { Grid, Group } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group);
var initial = true;
let grid: Grid = new Grid({
  dataSource: data,
  allowGrouping: true,
  groupSettings: { columns: ['ShipCity'] },
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
    { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
    { field: 'ShipCity', headerText: 'Ship City', width: 150 },
    { field: 'ShipName', headerText: 'Ship Name', width: 150 }
  ],
  dataBound: bound,
  height: 267
});
grid.appendTo('#Grid');
function bound() {
  if(initial === true) {
    grid.groupModule.collapseAll();
    initial = false;
  }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Merge duplicate cells in specific column and export in EJ2 JavaScript Grid control

You can merge the duplicate cells (based on the value) for the particular column of Grid by using '[dataBound](#)' event. At the same time, you can also merge the duplicate cells for particular column while exporting by using the [excelQueryCellInfo](#) event for Excel/CSV and [pdfQueryCellInfo](#) event for PDF exporting.

In the below demo, the duplicate cells are merged for the **OrderID** column in Grid view and its exporting.

INDEX.TS

```

import { Grid, ExcelQueryCellInfoEventArgs, ExcelCell,
PdfQueryCellInfoEventArgs, ExcelExport } from '@syncfusion/ej2-grids';
import { PdfExport, PdfGridCell, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(ExcelExport, PdfExport, Toolbar );
// global variable declaration for excel export
let gridcells: ExcelCell;
let ValOfOrderID:Number =null;
let i=1;
// global variable declaration for pdf Export
let pdfGridcell: PdfGridCell;
let ValOfOrderID_PDF :Number = null;
let pdfCellindex:Number =1;
let grid: Grid = new Grid(
    {
        dataSource: data,
        allowExcelExport:true,
        allowPdfExport:true,
        toolbar:["ExcelExport","PdfExport"],
        toolbarClick: toolbarClick,
        dataBound: onDataBound,
        excelQueryCellInfo:excelQueryCellInfo,
        excelExportComplete: excelExportComplete,
        pdfQueryCellInfo: pdfQueryCellInfo,
        pdfExportComplete: pdfExportComplete,
        allowPaging:true,
        pageSettings: { pageSize:5 },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', width: 120,
textAlign: 'Right' },
            { field: 'CustomerID', headerText: 'Customer ID', width: 130 },
            { field: "City", headerText:"ShipCity",width:120}
        ],
        height: 260,
    });
grid.appendTo('#Grid');
function toolbarClick(args){
    if (args.item.text === 'PDF Export') {
        this.pdfExport();
    }
    if (args.item.text === 'Excel Export') {
        this.excelExport();
    }
}
function onDataBound(args: any) {
let previousData: string = null;
let stRowIndex: number = null;
let endRowIndex: number = null;
let grid = this;
let rows = this.getRows();
let data = this.getCurrentViewRecords();
for (let i = 0, len = rows.length; i < len; i++) {
    if (!previousData) {
        previousData = data[i]['OrderID'];
        stRowIndex = parseInt(rows[i].getAttribute("aria-rowindex"));
    }
    else if (previousData === data[i]['OrderID']) {

```



```

        rows[i].children[0].classList.add('e-hide');
    }
    else if (previousData && previousData !== data[i]['OrderID']) {
        if (grid.getRows().length > 0 && grid.getRows().length >
stRowIndex) {
            endRowIndex = parseInt(rows[i].getAttribute("aria-
rowindex"), 10);
            let targetCell: Element[] =

[[]].slice.call(grid.getRows()[stRowIndex].querySelectorAll('.e-
rowcell')).filter((cell: Element) =>
                parseInt(cell.getAttribute('aria-colindex'), 10) ===
parseInt(rows[i].children[0].getAttribute('aria-colindex')));
            (targetCell[0] as any).setAttribute("rowSpan", endRowIndex -
stRowIndex);

            previousData = data[i]['OrderID'];
            stRowIndex = parseInt(rows[i].getAttribute("aria-rowindex"),
10);
        }
    }
    if (rows[i].children[0].classList.contains("e-hide") || i < len) {
        endRowIndex = parseInt(rows[i].getAttribute("aria-rowindex"),
10);

        if (endRowIndex > 0) {
            let targetCell: Element[] =
[[]].slice.call(grid.getRows()[stRowIndex].querySelectorAll('.e-
rowcell')).filter((cell: Element) =>
                parseInt(cell.getAttribute('aria-colindex'), 10) ===
parseInt(rows[i].children[0].getAttribute('aria-colindex')));
            (targetCell[0] as any).setAttribute("rowSpan", endRowIndex +
1);
        }
    }
}
}

function excelQueryCellInfo(args: ExcelQueryCellInfoEventArgs) {
    if (!ValOfOrderID && args.column.field == "OrderID") {
        ValOfOrderID = args.data["OrderID"];
        gridcells = (args.cell as ExcelCell);
    }
    else if (ValOfOrderID && args.column.field == "OrderID" && ValOfOrderID
== args.data["OrderID"]) {
        i++;
    } else if (ValOfOrderID !== args.data["OrderID"] && args.column.field ==
"OrderID") {
        (gridcells as ExcelCell).rowSpan = i;
        ValOfOrderID = args.data["OrderID"];
        gridcells = (args.cell as ExcelCell);
        i = 1;
    }
}

// Reset the excel export global variable values
function excelExportComplete(args: ExcelExportCompleteArgs) {
    ValOfOrderID = null;
    gridcells = null;
    i=1;
}

```

```
function pdfQueryCellInfo(args: PdfQueryCellInfoEventArgs) {
    if(!ValOfOrderID_PDF && args.column.field == "OrderID"){
        ValOfOrderID_PDF = args.data["OrderID"];
        pdfGridcell = (args.cell as PdfGridCell);
    }
    else if(ValOfOrderID_PDF && args.column.field == "OrderID" &&
ValOfOrderID_PDF ==args.data["OrderID"]){
        (pdfCellindex as any)++;
    } else if(ValOfOrderID_PDF != args.data["OrderID"] &&
args.column.field == "OrderID") {
        (pdfGridcell as PdfGridCell).rowSpan = pdfCellindex as any;
        ValOfOrderID_PDF = args.data["OrderID"];
        pdfGridcell=(args.cell as PdfGridCell);
        pdfCellindex = 1 ;
    }
}
// Reset the pdf export global variable values
function pdfExportComplete(args: PdfExportCompleteArgs) {
    ValOfOrderID_PDF=null;
    pdfGridcell=null;
    pdfCellindex =1;
}
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Get row cell index in EJ2 JavaScript Grid control

You can get the specific row and cell index of the grid by using `rowSelected` event of the grid. Here, we can get the row and cell index by using `aria-rowindex`(get row Index from `tr` element) and `aria-colindex`(column index from `td` element) attribute.

INDEX.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { data } from './datasource.ts';
import { DataManager } from '@syncfusion/ej2-data';
Grid.Inject(Page);
let grid: Grid = new Grid(
    {
        dataSource: data,
        allowPaging: true,
        pageSettings: { pageCount: 5, pageSize: 5 },
        columns: [
            { field: 'OrderID', headerText: 'Order ID', width: 120,
textAlign: 'Right' },
            { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
            { field: 'OrderDate', headerText: 'Order Date', width: 130,
format: 'yMd', textAlign: 'Right' },
            { field: 'Freight', width: 120, format: 'C2', textAlign: 'Right'
},
            { field: 'ShipCountry', visible: false, headerText: 'Ship
Country', width: 150 }
        ],
        rowSelected: rowSelected
    });
grid.appendTo('#Grid');

```

```
function rowSelected(args: any): void {
    alert("row index: "+args.row.getAttribute('aria-rowindex'));
    alert("column index: "+args.target.getAttribute('aria-colindex'));
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;
        }
        .e-grid.e-gridhover tr[role='row']:hover {
            background-color: #eee;
        }
        .e-expand::before {
            content: '\e5b8';
        }
    </style>
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
```

```

</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Display null values at bottom in EJ2 JavaScript Grid control

By default the null values are displayed at bottom of the Grid row while perform sorting in ascending order. As well as this values are displayed at top of the Grid row while perform sorting with descending order. But you can customize this default order to display the null values at always bottom row of the Grid by using [column.sortComparer](#) method.

In the below demo we have displayed the null values at bottom of the Grid row while sorting the **OrderDate** column in both ways.

INDEX.TS

```

import { Grid, Sort, Page, SortEventArgs } from '@syncfusion/ej2-grids';
import { OrderData } from './datasource.ts';
Grid.Inject(Sort, Page);
let grid: Grid = new Grid({
    dataSource: OrderData,
    allowSorting: true,
    actionBegin: actionBegin,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', width: 100 },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'OrderDate', headerText: 'Order Date', format: 'yMd',
sortComparer: sortComparer, width: 120},
        { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
    ],
    height: 265
});
let action: string;
function actionBegin(args: SortEventArgs) {
    if (args.requestType === 'sorting') {
        action = args.direction;
    }
}
function sortComparer(reference, comparer) {
    const sortAsc = action === 'Ascending' ? true : false;

```

```

        if (sortAsc && reference === null) {
            return 1;
        } else if (sortAsc && comparer === null) {
            return -1;
        } else if (!sortAsc && reference === null) {
            return -1;
        } else if (!sortAsc && comparer === null) {
            return 1;
        } else {
            return reference - comparer;
        }
    }
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>

  <div id="container">
    <div id="dropDown"></div>
    <div id="Grid"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Enable editing in single click in EJ2 JavaScript Grid control

Normal Editing

You can make a row editable on a single click with **Normal** mode of editing in Grid, by using the [startEdit](#) and [endEdit](#) methods.

Bind the **mouseup** event for Grid and in the event handler call the [startEdit](#) and [endEdit](#), based on the clicked target element.

INDEX.TS

```
import { Grid, Edit, IGrid, Toolbar } from '@syncfusion/ej2-grids';
import { MouseEventArgs } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let grid: Grid = new Grid({
  dataSource: data,
  editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
  toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
    { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
    { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
    { field: 'ShipCountry', headerText: 'Ship Country', width: 150 }
  ],
  load: () => {
    let instance: IGrid =
document.getElementById('Grid').ej2_instances[0];
    instance.element.addEventListener('mouseup', (e: MouseEventArgs) =>
{
      if (e.target.classList.contains("e-rowcell")) {
        if (instance.isEdit)
          instance.endEdit();
        let index: number =
parseInt(e.target.getAttribute("Index"));
        instance.selectRow(index);
        instance.startEdit();
      }
    })
  })
});
```

```

    }
    height: 220
  });
  grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
      background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
      background-color: #eee;
    }
    .e-expand::before {
      content: '\e5b8';
    }
    .empImage {
      margin: 6px 16px;
      float: left;
      width: 50px;
      height: 50px;
    }
  </style>

```



```

</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Open dropdown edit popup on single click

You can open the default dropdown edit popup with single click edit by focusing the dropdown element and calling the EJ2 dropdown list's [showPopup](#) method in the Grid's [actionComplete](#) event. In this demo we have used a global flag variable in the 'mouseup' event to ensure if the dropdown column is the clicked edit target.

INDEX.TS

```

import { Grid, Edit, IGrid, Toolbar } from '@syncfusion/ej2-grids';
import { MouseEventArgs } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
Grid.Inject(Edit, Toolbar);
let isDropdown = false;
let grid: Grid = new Grid({
    dataSource: data,
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100, isPrimaryKey: true },
        { field: 'CustomerID', headerText: 'Customer ID', width: 120 },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150 }
    ],
    actionComplete: onActionComplete,
    load: () => {
        let instance: IGrid =
document.getElementById('Grid').ej2_instances[0];
        instance.element.addEventListener('mouseup', (e: MouseEventArgs) =>
{
            if (e.target.classList.contains("e-rowcell")) {
                if (instance.isEdit)

```

```

        instance.endEdit();
        let rowInfo = instance.getRowInfo(e.target);
        if (rowInfo.column.field === "ShipCountry")
            isDropdown = true;
        instance.selectRow(rowInfo.rowIndex);
        instance.startEdit();
    }
    })
    },
    height: 220
});
grid.appendTo('#Grid');
function onActionComplete(args) {
    if (args.requestType === "beginEdit" && isDropdown) {
        isDropdown = false;
        let dropdownObj = args.form.querySelector('.e-
dropdownlist').ej2_instances[0];
        dropdownObj.element.focus();
        dropdownObj.showPopup();
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-row[aria-selected="true"] .e-customizedExpandcell {
        background-color: #e0e0e0;
    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid print in EJ2 JavaScript Grid control

You can add your title in the header through an [beforePrint](#) event. With the help of this event you can add your title element as you want.

INDEX.TS

```

import { Grid, Page, Toolbar } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page, Toolbar);
let grid: Grid = new Grid({
    dataSource: data,
    allowPaging: true,
    beforePrint: beforePrint,
    toolbar: ['Print'],
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120, type: 'number' },

```

```

        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
width: 120, format: 'C2' },
        { field: 'OrderDate', headerText: 'Order Date', textAlign: 'Right',
width: 140, format: 'yMd' }
    ],
    height: 220
});
grid.appendTo('#Grid');
function beforePrint(args) {
    var div = document.createElement("Div")
    div.innerHTML = "Title here"
    div.style.textAlign = 'center';
    div.style.color = 'red';
    div.style.padding = '10px 0';
    div.style.fontSize = '25px';
    args.element.insertBefore(div, args.element.childNodes[0]);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change hierarchy grid icon in EJ2 JavaScript Grid control

You can change default expand/collapse icon of the Hierarchy Grid by using custom Css.

You can change the content code of the corresponding icon which you want.

To use hierarchical binding, inject the [DetailRow](#) module in the grid.

Css:

Use the below css in your `index.html` file.

```

,

.e-grid .e-icon-grightarrow::before,
.e-grid-menu .e-icon-grightarrow::before {
content: '\e7f9';
}

.e-grid .e-icon-gdownarrow::before,
.e-grid-menu .e-icon-gdownarrow::before {
content: '\e934';
}
,

```

In this below demo, we have changed the expand/collapse icon to Plus/Minus icon.

INDEX.TS

```

import { Grid, DetailRow } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
import { employeeData } from './employeeData.ts';
Grid.Inject(DetailRow);
let grid: Grid = new Grid({

```

```

        dataSource: employeeData,
        columns: [
            { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
'Right', width: 120 },
            { field: 'FirstName', headerText: 'First Name', width: 150 },
            { field: 'City', headerText: 'City', width: 150 },
            { field: 'Country', headerText: 'Country', width: 150 }
        ],
        childGrid: {
            dataSource: data,
            queryString: 'EmployeeID',
            columns: [
                { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 120 },
                { field: 'CustomerID', headerText: 'Customer ID', width: 150 },
                { field: 'ShipCity', headerText: 'Ship City', width: 150 },
                { field: 'ShipName', headerText: 'Ship Name', width: 150 }
            ],
        },
        height: 315
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<style>
    .e-grid .e-icon-grightarrow::before,
    .e-grid-menu .e-icon-grightarrow::before {
        content: '\e7f9';
    }

    .e-grid .e-icon-gdownarrow::before,
    .e-grid-menu .e-icon-gdownarrow::before {
        content: '\e934';
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Render image in drop down list in EJ2 JavaScript Grid control

To display an image in the DropDownList editor component, you can utilize the `itemTemplate` property. This property allows you to customize the content of each item in the dropdown list.

The below example shows how to use the `itemTemplate` property in the DropDownList editor component to display an image in the dropdown list. Here, we render the image for each item in the dropdown list of the foreign key column `Employee Name`.

INDEX.TS

```

import { Grid, ForeignKey, Edit, Toolbar, Page } from '@syncfusion/ej2-
grids';
import { data } from './datasource.ts';
import { employeeData } from './employeeData.ts';
Grid.Inject(ForeignKey, Edit, Toolbar, Page);
let grid: Grid = new Grid(
    {
        dataSource: data,
        allowPaging: true,
        toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
        editSettings: { allowEditing: true, allowAdding: true,
allowDeleting: true },

```

```

        columns: [
            { field: 'OrderID', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
            { field: 'EmployeeID', headerText: 'Employee Name', width: 150,
foreignKeyValue: 'FirstName',dataSource: employeeData,
editType:'dropdownedit', edit: { params: { itemTemplate:'<div>' + '<div class="ename"> ${FirstName} </div></div>', }, },
            { field: 'Freight', headerText: 'Freight', width: 100,
textAlign: 'Right'},
            { field: 'ShipName', headerText: 'Ship Name', width: 180 }
        ],
        height: 315
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <style>
        .e-row[aria-selected="true"] .e-customizedExpandcell {
            background-color: #e0e0e0;

```



```

    }
    .e-grid.e-gridhover tr[role='row']:hover {
        background-color: #eee;
    }
    .e-expand::before {
        content: '\e5b8';
    }
    .empImage {
        margin: 6px 16px;
        float: left;
        width: 50px;
        height: 50px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the Empty Record Template in EJ2 JavaScript Grid control

The empty record template feature in the Grid allows you to use custom content such as images, text, or other components, when the grid doesn't contain any records to display. This feature replaces the default message of 'No records to display' typically shown in the grid.

To activate this feature, set the `emptyRecordTemplate` property of the Grid. The `emptyRecordTemplate` property expects the HTML element or a function that returns the HTML element.

In the following example, an image and text have been rendered as a template to indicate that the grid has no data to display.

INDEX.TS

```

import { Grid, Page, Selection, Toolbar, Edit } from '@syncfusion/ej2-
grids';
import { data } from './datasource.ts';
import { DataManager } from '@syncfusion/ej2-data';
Grid.Inject(Page, Selection, Toolbar, Edit );
let dropDownDataSource = new DataManager(data);
let grid: Grid = new Grid(
    {

```

```

        dataSource: [],
        allowPaging: true,
        toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
        emptyRecordTemplate: '#emptytemplate',
        editSettings: { allowEditing: true, allowAdding: true,
allowDeleting: true},
        columns: [
            { field: 'OrderID', isPrimaryKey: true, headerText: 'Order ID',
textAlign: 'Right', validationRules: { required: true, number: true },
width: 140 },
            { field: 'CustomerID', headerText: 'Customer ID',
validationRules: { required: true }, width: 140 },
            { field: 'Freight', headerText: 'Freight', textAlign: 'Right',
editType: 'numericedit', width: 140, format: 'C2', validationRules: {
required: true } },
            { field: 'OrderDate', headerText: 'Order Date', editType:
'datetimepickeredit', width: 160, format: { type: 'dateTime', format: 'M/d/y
hh:mm a' }, },
            { field: 'ShipCountry', headerText: 'Ship Country', editType:
'dropdownedit', width: 150, edit: { params: { dataSource: dropDownDataSource
, fields: {text:"ShipCountry",value:"ShipCountry"}}}}
        ],
        pageSettings: { pageCount: 5 }
    });
    grid.appendTo('#Grid');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<style>
    .emptyRecordTemplate {
        text-align: center;
    }
    .e-emptyRecord {
        display: block;
        margin: 10px auto;
    }
</style>
<script id="emptytemplate" type="text/x-template">
    <div class='emptyRecordTemplate'>
        
        <span>There is no data available to display at the
moment.</span>
    </div>
</script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Grid"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

HeatMap Chart

Getting started in EJ2 JavaScript Heatmap chart control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

control Initialization

The Essential JS 2 JavaScript controls can be initialized by using either of the following ways.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

Using local script and style references in a HTML page

Step 1: Create an app folder `myapp` for Essential JS 2 JavaScript controls.

Step 2: You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

Syntax:

Script: `(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js`

Example:

Script: `C:/Program Files (x86)/Syncfusion/Essential Studio/16.2.45/Essential JS 2/ej2-heatmap/dist/global/ej2-heatmap.min.js`

Step 3: Create a folder `myapp/resources` and copy/paste the dependent scripts and styles from the above installed location to `myapp/resources` location.

Step 4: Create a HTML page (`index.html`) in `myapp` location and add the Essentials JS 2 script and style references.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<script src="resources/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/ej2-svg-base.min.js" type="text/javascript"></script>
<script src="resources/ej2-heatmap.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Step 5: Now, add the `div` element and initiate the `Essential JS 2 Heatmap` control in the `index.html` by using following code

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
```

```

<script src="resources/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/ej2-svg-base.min.js" type="text/javascript"></script>
<script src="resources/ej2-heatmap.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element -->
<div id="container"></div>
<script>
// initialize heatmap control
var heatmap = new ej.heatmap.HeatMap();
// Render initialized heatmap.
heatmap.appendTo('#container')
</script>
</body>
</html>

```

Step 6: Now, run the **index.html** in web browser, it will render the **Essential JS 2 HeatMap** control.

Using CDN link for script and style reference

Step 1: Create an app folder **quickstart** for the Essential JS 2 JavaScript controls.

Step 2: The Essential JS 2 controls' global scripts are already hosted in the below CDN link formats.

Common Control Script:

Syntax: https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/dist/ej2.min.js

Example:

Script: <https://cdn.syncfusion.com/ej2/23.1.36/dist/ej2.min.js>

or

Individual Control Scripts:

Syntax

http://cdn.syncfusion.com/ej2/{RELEASEVERSION}/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js

Example:

HeatMap Script: <https://cdn.syncfusion.com/ej2/23.1.36/ej2-heatmap/dist/global/ej2-heatmap.min.js>

Dependency Scripts

<https://cdn.syncfusion.com/ej2/23.1.36/ej2-base/dist/global/ej2-base.min.js>

<https://cdn.syncfusion.com/ej2/23.1.36/ej2-data/dist/global/ej2-data.min.js>

<https://cdn.syncfusion.com/ej2/23.1.36/ej2-svg-base/dist/global/ej2-svg-base.min.js>

Step 3: Create a HTML page (index.html) in quickstart location and add the CDN link references. Now, add the `div` element and initiate the **Essential JS 2 Heatmap** control in the index.html by using following code.

INDEX.HTML

```
<!DOCTYPE html><html xmlns="http://www.w3.org/1999/xhtml"><head>
  <title>Essential JS 2</title>
  <script src="http://cdn.syncfusion.com/ej2/ej2-
base/dist/global/ej2-base.min.js" type="text/javascript"></script>
  <script src="http://cdn.syncfusion.com/ej2/ej2-
data/dist/global/ej2-data.min.js" type="text/javascript"></script>
  <script src="http://cdn.syncfusion.com/ej2/ej2-svg-
base/dist/global/ej2-svg-base.min.js" type="text/javascript"></script>
  <script src="http://cdn.syncfusion.com/ej2/ej2-
heatmap/dist/global/ej2-heatmap.min.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
  <body>
    <!-- Add the HTML <div> element -->
    <div id="container"></div>
    <script>
      // initialize heatmap component
      var heatmap = new ej.heatmap.HeatMap();
      // Render initialized heatmap.
      heatmap.appendTo('#container')
    </script>

    <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Step 4: Now, run the **index.html** in web browser, it will render the **Essential JS 2 HeatMap** control.

Populate HeatMap with data

This section explains how to populate the following two-dimensional array data to the HeatMap.

INDEX.JS

```
var heatmapData = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
var heatmap = new ej.heatmap.HeatMap({
    dataSource: heatmapData
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Enable axis labels

You can add axis labels to the HeatMap and format those labels using the [xAxis](#) and [yAxis](#) properties. Axis labels provide additional information about the data points populated in the HeatMap.

INDEX.JS

```

var heatmapData = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
var heatmap = new ej.heatmap.HeatMap({
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul',
            'Karin', 'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


Add HeatMap title

Add a title using the [titleSettings](#) property to the HeatMap to provide quick information to the user about the data populated in the HeatMap.

INDEX.JS

```
var heatmapData = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
var heatmap = new ej.heatmap.HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul',
            'Karin', 'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable legend

Use a legend for the HeatMap in the [legendSettings](#) object by setting the [visible](#) property to **true** and injecting the **Legend** module using the **HeatMap.Inject(Legend)** method.

INDEX.JS

```

var heatmapData = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
var heatmap = new ej.heatmap.HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul',
            'Karin', 'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
    legendSettings: {

```

```

        position: 'Right',
        showLabel: true,
        height: "150"
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Add data label

Add data labels to improve the readability of the HeatMap. This can be achieved by setting the [showLabel](#) property to **true** in the [cellSettings](#) object.

INDEX.JS

```

var heatmapData = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];

```

```

var heatmap = new ej.heatmap.HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  cellSettings: {
    showLabel: true,
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
      'Michael', 'Robert', 'Laura', 'Anne', 'Paul',
      'Karin', 'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  dataSource: heatmapData,
  legendSettings: {
    position: 'Right',
    showLabel: true,
    height: "150"
  }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Add custom cell palette

The default palette settings of the HeatMap cells can be customized by using the [paletteSettings](#) property.

Using the [palette](#) property in `paletteSettings` object, you can change the color set for the cells.

You can change the color mode of the cells to fixed or gradient mode using the [type](#) property.

INDEX.JS

```
var heatmapData = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
var heatmap = new ej.heatmap.HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  cellSettings: {
    showLabel: true,
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
      'Michael', 'Robert', 'Laura', 'Anne', 'Paul',
      'Karin', 'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  dataSource: heatmapData,
  legendSettings: {
    position: 'Right',
    showLabel: true,
    height: "150"
  },
  paletteSettings: {
    palette: [{ value: 0, color: '#C06C84' },
      { value: 50, color: '#6C5B7B' }],
  }
});
```

```

        { value: 100, color: '#355C7D' }
    ],
    type: "Gradient"
},
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable tooltip

The tooltip is used when you cannot display information by using the data labels due to space constraints. You can enable the tooltip by setting the [showTooltip](#) property to **true** and injecting the **Tooltip** module using the **HeatMap.Inject(Tooltip)** method.

INDEX.JS

```

var heatmapData = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],

```

```

[74, 33, 88, 23, 86, 59]];
var heatmap = new ej.heatmap.HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  cellSettings: {
    showLabel: true,
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
      'Michael', 'Robert', 'Laura', 'Anne', 'Paul',
      'Karin', 'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  dataSource: heatmapData,
  legendSettings: {
    position: 'Right',
    showLabel: true,
    height: "150"
  },
  showTooltip: true,
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Working with data in EJ2 JavaScript Heatmap chart control

Heat map visualizes the JSON data and two-dimensional array data. Using the data adaptor support, data can be bound to the heat map.

Data adaptor

Heat map supports the following types of data binding with the adaptor support.

- Array
- Table binding
- Cell binding
- JSON data
- Table binding
- Cell binding

Array - table binding

This data type is a collection of one dimensional array objects, at which each inner array contains data points for an X-axis data label.

This is the default data binding type for heat map. You can also directly bind the array object to the [dataSource](#) property.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, ITooltipEventArgs } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip);
let heatmapData: Object = [
  [9.5, 2.2, 4.2, 8.2, -0.5, 3.2, 5.4, 7.4, 6.2, 1.4],
  [4.3, 8.9, 10.8, 6.5, 5.1, 6.2, 7.6, 7.5, 6.1, 7.6],
  [3.9, 2.7, 2.5, 3.7, 2.6, 5.1, 5.8, 2.9, 4.5, 5.1],
  [2.4, -3.7, 4.1, 6.0, 5.0, 2.4, 3.3, 4.6, 4.3, 2.7],
  [2.0, 7.0, -4.1, 8.9, 2.7, 5.9, 5.6, 1.9, -1.7, 2.9],
  [5.4, 1.1, 6.9, 4.5, 2.9, 3.4, 1.5, -2.8, -4.6, 1.2],
  [-1.3, 3.9, 3.5, 6.6, 5.2, 7.7, 1.4, -3.6, 6.6, 4.3],
  [-1.6, 2.3, 2.9, -2.5, 1.3, 4.9, 10.1, 3.2, 4.8, 2.0],
  [10.8, -1.6, 4.0, 6.0, 7.7, 2.6, 5.6, -2.5, 3.8, -1.9],
  [6.2, 9.8, -1.5, 2.0, -1.5, 4.3, 6.7, 3.8, -1.2, 2.4],
  [1.2, 10.9, 4.0, -1.4, 2.2, 1.6, -2.6, 2.3, 1.7, 2.4],
  [5.1, -2.4, 8.2, -1.1, 3.5, 6.0, -1.3, 7.2, 9.0, 4.2]
];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'GDP Growth Rate for Major Economies (in Percentage)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  }
});

```



```

    },
    xAxis: {
        labels: ['China', 'India', 'Australia', 'Mexico', 'Canada',
        'Brazil',
        'USA', 'UK', 'Germany', 'Russia', 'France', 'Japan'],
        labelRotation: 45,
        labelIntersectAction: 'None',
    },
    yAxis: {
        labels: ['2008', '2009', '2010', '2011', '2012', '2013', '2014',
        '2015', '2016', '2017']
    }, paletteSettings: {
        palette: [
            { value: -1, color: '#F0D6AD' },
            { value: 0, color: '#9da49a' },
            { value: 3.5, color: '#d7c7a7' },
            { value: 6.0, color: '#6e888f' },
            { value: 7.5, color: '#466f86' },
            { value: 10, color: '#19547B' },
        ],
    },
    legendSettings: {
        visible: false
    },
    tooltipRender: (args: ITooltipEventArgs) => {
        args.content = [args.yLabel + ' | ' + args.xLabel + ' : ' +
        args.value + ' %'];
    },
    dataSource: heatmapData
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Array - cell binding

This data type is a collection of array objects that contain information about the row index, column index, and data value for each cell. You can bind the data to heat map by using the [data](#) property in the [dataSource](#) and setting the [adaptorType](#) property to **Cell**.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, Adaptor, ITooltipEventArgs } from
'@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: Object = [
    [0, 0, 10.75], [0, 1, 14.5], [0, 2, 25.5], [0, 3, 39.5], [0, 4,
59.75], [0, 5, 35.50], [0, 6, 75.5],
    [1, 0, 20.75], [1, 1, 35.5], [1, 2, 29.5], [1, 3, 75.5], [1, 4, 80],
[1, 5, 65], [1, 6, 85],
    [2, 0, 6], [2, 1, 18.5], [2, 2, 30.05], [2, 3, 35.5], [2, 4, 40.75],
[2, 5, 50.75], [2, 6, 65],
    [3, 0, 30.5], [3, 1, 20.5], [3, 2, 45.30], [3, 3, 50], [3, 4, 55],
[3, 5, 85.80], [3, 6, 87.5],
    [4, 0, 10.5], [4, 1, 20.75], [4, 2, 35.5], [4, 3, 35.5], [4, 4,
45.5], [4, 5, 65.], [4, 6, 75.5],
    [5, 0, 45.5], [5, 1, 20.75], [5, 2, 45.5], [5, 3, 50.75], [5, 4,
79.30], [5, 5, 84.20], [5, 6, 87.36],
    [6, 0, 26.82], [6, 1, 70], [6, 2, 75], [6, 3, 79.5], [6, 4, 88.5],
[6, 5, 89.5], [6, 6, 91.75],
    [7, 0, 15.75], [7, 1, 20.75], [7, 2, 25.5], [7, 3, 42.35], [7, 4,
45.15], [7, 5, 76.5], [7, 6, 80.5],
    [8, 0, 1.98], [8, 1, 15.23], [8, 2, 43], [8, 3, 49], [8, 4, 63.80],
[8, 5, 67.97], [8, 6, 70.52],
    [9, 0, 14.31], [9, 1, 42.87], [9, 2, 77.28], [9, 3, 77.82], [9, 4,
81.44], [9, 5, 81.92], [9, 6, 83.75],
    [10, 0, 25.5], [10, 1, 35.5], [10, 2, 40.5], [10, 3, 45.05], [10, 4,
50.5], [10, 5, 75.5], [10, 6, 90.58]
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Percentage of Individuals Using Internet by Country',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['China', 'Australia', 'Mexico', 'Canada', 'Brazil',
'USA',
            'UK', 'Germany', 'Russia', 'France', 'Japan'],
    },

```

```

        yAxis: {
            labels: ['2000', '2005', '2010', '2011', '2012', '2013',
'2014'],
        },
        dataSource: heatmapData,
        dataSourceSettings: {
            isJsonData: false,
            adaptorType: 'Cell'
        },
        paletteSettings: {
            palette: [{ color: '#3498DB' },
            { color: '#2C3E50' }
        ]
        },
        cellSettings: {
            border: {
                width: '0'
            },
            textStyle: {
                color: 'white'
            },
            format: '{value} %'
        },
        legendSettings: {
            visible: false,
        },
        tooltipRender: (args: ITooltipEventArgs) => {
            args.content = [args.yLabel + ' | ' + args.xLabel + ' : ' +
args.value + ' %'];
        }
    });
    heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

JSON data - table binding

In JSON table data binding, each JSON object contains an X-axis data point as row header and all the corresponding Y-axis data values.

You can bind the JSON table data to the heat map using the [data](#) property in [dataSource](#). To achieve this, you should enable the [isJsonData](#) property and define the [adaptorType](#) property as **Table**. The [xDataMapping](#) property is used to map the row header in JSON data.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, Adaptor} from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: object = [
  { 'Region': 'USA', '2000': 93, '2004': 101, '2008': 112, '2012': 103, '2016': 121 },
  { 'Region': 'GBR', '2000': 28, '2004': 30, '2008': 49, '2012': 65, '2016': 67 },
  { 'Region': 'China', '2000': 58, '2004': 63, '2008': 100, '2012': 91, '2016': 70 },
  { 'Region': 'Russia', '2000': 89, '2004': 90, '2008': 60, '2012': 69, '2016': 55 },
  { 'Region': 'Germany', '2000': 56, '2004': 49, '2008': 41, '2012': 44, '2016': 42 },
  { 'Region': 'Japan', '2000': 18, '2004': 37, '2008': 25, '2012': 38, '2016': 41 },
  { 'Region': 'France', '2000': 38, '2004': 33, '2008': 43, '2012': 35, '2016': 42 },
  { 'Region': 'KOR', '2000': 28, '2004': 30, '2008': 32, '2012': 30, '2016': 21 },
  { 'Region': 'Italy', '2000': 34, '2004': 32, '2008': 27, '2012': 28, '2016': 28 }];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Olympic Medal Achievements of most Successful Countries',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['China', 'France', 'GBR', 'Germany', 'Italy', 'Japan', 'KOR', 'Russia', 'USA'],
    labelRotation: 45,
    labelIntersectAction: 'None',
  },
  yAxis: {
    title : {text: 'Olympic Year'},
    labels: ['2000', '2004', '2008', '2012', '2016'],
  }
});

```

```

    },
    dataSource: heatmapData,
    dataSourceSettings: {
        isJsonData: true,
        adaptorType: 'Table',
        xDataMapping: 'Region',
    },
    paletteSettings: {
        palette: [{ color: '#F0C27B' },
        { color: '#4B1248' }
        ],
    },
    cellSettings: {
        border: {
            width: 1,
            radius: 4,
            color: 'white'
        }
    }
},
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

JSON data - Cell binding

In JSON cell data binding, each JSON object consists a value for each cell along with a mapping value for row and column. You can bind the JSON cell data having information for each cell to the heat map using the [data](#) property in [dataSource](#). To achieve this, you should define the [adaptorType](#) property as **Cell**,

and enable the [isJsonData](#) property. Now, map the fields of data by using the [valueMapping](#), [xDataMapping](#) and [yDataMapping](#) properties.

INDEX.TS

```
import { HeatMap, Legend, Tooltip, Adaptor} from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: object = [
    { 'rowid': 'France', 'columnid': '2010', 'value': '77.6' },
    { 'rowid': 'France', 'columnid': '2011', 'value': '79.4' },
    { 'rowid': 'France', 'columnid': '2012', 'value': '80.8' },
    { 'rowid': 'France', 'columnid': '2013', 'value': '86.6' },
    { 'rowid': 'France', 'columnid': '2014', 'value': '83.7' },
    { 'rowid': 'France', 'columnid': '2015', 'value': '84.5' },
    { 'rowid': 'France', 'columnid': '2016', 'value': '82.6' },
    { 'rowid': 'USA', 'columnid': '2010', 'value': '60.6' },
    { 'rowid': 'USA', 'columnid': '2011', 'value': '65.4' },
    { 'rowid': 'USA', 'columnid': '2012', 'value': '70.8' },
    { 'rowid': 'USA', 'columnid': '2013', 'value': '73.8' },
    { 'rowid': 'USA', 'columnid': '2014', 'value': '75.3' },
    { 'rowid': 'USA', 'columnid': '2015', 'value': '77.5' },
    { 'rowid': 'USA', 'columnid': '2016', 'value': '77.6' },
    { 'rowid': 'Spain', 'columnid': '2010', 'value': '64.9' },
    { 'rowid': 'Spain', 'columnid': '2011', 'value': '52.6' },
    { 'rowid': 'Spain', 'columnid': '2012', 'value': '60.8' },
    { 'rowid': 'Spain', 'columnid': '2013', 'value': '65.6' },
    { 'rowid': 'Spain', 'columnid': '2014', 'value': '52.6' },
    { 'rowid': 'Spain', 'columnid': '2015', 'value': '68.5' },
    { 'rowid': 'Spain', 'columnid': '2016', 'value': '75.6' },
    { 'rowid': 'China', 'columnid': '2010', 'value': '55.6' },
    { 'rowid': 'China', 'columnid': '2011', 'value': '52.3' },
    { 'rowid': 'China', 'columnid': '2012', 'value': '54.8' },
    { 'rowid': 'China', 'columnid': '2013', 'value': '51.1' },
    { 'rowid': 'China', 'columnid': '2014', 'value': '55.6' },
    { 'rowid': 'China', 'columnid': '2015', 'value': '56.9' },
    { 'rowid': 'China', 'columnid': '2016', 'value': '59.3' },
    { 'rowid': 'Italy', 'columnid': '2010', 'value': '43.6' },
    { 'rowid': 'Italy', 'columnid': '2011', 'value': '43.2' },
    { 'rowid': 'Italy', 'columnid': '2012', 'value': '55.8' },
    { 'rowid': 'Italy', 'columnid': '2013', 'value': '50.1' },
    { 'rowid': 'Italy', 'columnid': '2014', 'value': '48.5' },
    { 'rowid': 'Italy', 'columnid': '2015', 'value': '50.7' },
    { 'rowid': 'Italy', 'columnid': '2016', 'value': '52.4' },
    { 'rowid': 'UK', 'columnid': '2010', 'value': '28.2' },
    { 'rowid': 'UK', 'columnid': '2011', 'value': '31.6' },
    { 'rowid': 'UK', 'columnid': '2012', 'value': '29.8' },
    { 'rowid': 'UK', 'columnid': '2013', 'value': '33.1' },
    { 'rowid': 'UK', 'columnid': '2014', 'value': '32.6' },
    { 'rowid': 'UK', 'columnid': '2015', 'value': '34.4' },
    { 'rowid': 'UK', 'columnid': '2016', 'value': '35.8' },
    { 'rowid': 'Germany', 'columnid': '2010', 'value': '26.8' },
    { 'rowid': 'Germany', 'columnid': '2011', 'value': '29' },
    { 'rowid': 'Germany', 'columnid': '2012', 'value': '26.8' },
    { 'rowid': 'Germany', 'columnid': '2013', 'value': '27.6' },
    { 'rowid': 'Germany', 'columnid': '2014', 'value': '33' },
    { 'rowid': 'Germany', 'columnid': '2015', 'value': '35' },
    { 'rowid': 'Germany', 'columnid': '2016', 'value': '35.6' },

```

```

    { 'rowid': 'Mexico', 'columnid': '2010', 'value': '23.2' },
    { 'rowid': 'Mexico', 'columnid': '2011', 'value': '24.9' },
    { 'rowid': 'Mexico', 'columnid': '2012', 'value': '30.1' },
    { 'rowid': 'Mexico', 'columnid': '2013', 'value': '22.2' },
    { 'rowid': 'Mexico', 'columnid': '2014', 'value': '29.3' },
    { 'rowid': 'Mexico', 'columnid': '2015', 'value': '32.1' },
    { 'rowid': 'Mexico', 'columnid': '2016', 'value': '35' },
    { 'rowid': 'Thailand', 'columnid': '2010', 'value': '15.9' },
    { 'rowid': 'Thailand', 'columnid': '2011', 'value': '19.8' },
    { 'rowid': 'Thailand', 'columnid': '2012', 'value': '21.8' },
    { 'rowid': 'Thailand', 'columnid': '2013', 'value': '23.5' },
    { 'rowid': 'Thailand', 'columnid': '2014', 'value': '24.8' },
    { 'rowid': 'Thailand', 'columnid': '2015', 'value': '29.9' },
    { 'rowid': 'Thailand', 'columnid': '2016', 'value': '32.6' },
    { 'rowid': 'Austria', 'columnid': '2010', 'value': '22' },
    { 'rowid': 'Austria', 'columnid': '2011', 'value': '21.3' },
    { 'rowid': 'Austria', 'columnid': '2012', 'value': '24.2' },
    { 'rowid': 'Austria', 'columnid': '2013', 'value': '23.2' },
    { 'rowid': 'Austria', 'columnid': '2014', 'value': '25' },
    { 'rowid': 'Austria', 'columnid': '2015', 'value': '26.7' },
    { 'rowid': 'Austria', 'columnid': '2016', 'value': '28.1' },
  ];
  let heatmap: HeatMap = new HeatMap({
    titleSettings: {
      text: 'Most Visited Destinations by International Tourist Arrivals',
      textStyle: {
        size: '15px',
        fontWeight: '500',
        fontStyle: 'Normal',
        fontFamily: 'Segoe UI'
      }
    },
    xAxis: {
      labels: ['Austria', 'China', 'France', 'Germany', 'Italy', 'Mexico', 'Spain', 'Thailand', 'UK', 'USA'],
    },
    yAxis: {
      labels: ['2010', '2011', '2012', '2013', '2014', '2015', '2016'],
    },
    dataSource: heatmapData,
    dataSourceSettings: {
      isJsonData: true,
      adaptorType: 'Cell',
      xDataMapping: 'rowid',
      yDataMapping: 'columnid',
      valueMapping: 'value'
    },
    cellSettings: {
      border: {
        radius: 4,
        width: 1,
        color: 'white'
      },
      showLabel: true,
      format: '{value} M',
    },
    paletteSettings: {

```

```

        palette: [{ color: '#DCD57E' },
        { color: '#A6DC7E' },
        { color: '#7EDCA2' },
        { color: '#6EB5D0' }
        ],
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Empty points

The data points that use the **null** or **undefined** or empty string as value are considered as empty points. Empty data points are ignored and not displayed in the heat map, and these points are rendered with default palette. You can customize the empty data point color value using the [emptyPointColor](#) property.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, Adaptor } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, null, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],

```



```

        [41, 55, 73, 23, "", 79],
        [56, 69, 21, 86, 3, 33],
        [45, 7, undefined, 81, 95, 79],
        [60, 77, 74, 68, 88, 51],
        [25, 25, 10, 12, 78, 14],
        [25, 56, 55, 58, 12, 82],
        [74, 33, 88, 23, 86, 59]
    ];
    let heatmap: HeatMap = new HeatMap({
        xAxis: {
            valueType: "DateTime",
            minimum: new Date(2007, 0, 1),
            intervalType: "Years",
            labelFormat: "yyyy",
        }, yAxis: {
            valueType: "Numeric"
        },
        legendSettings: {
            visible: false,
        },
        dataSource: heatmapData
    });
    heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Binding nested JSON data

In complex data binding, you can bind the nested JSON data to the data points in the heat map. The nested data can be mapped using the [xDataMapping](#), [yDataMapping](#), [valueMapping](#) and [bubbleDataMapping](#) properties as string value concatenated by a dot.

INDEX.TS

```
import { HeatMap, Legend, Tooltip, Adaptor} from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: object = [
  { 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2010' }, 'data':{ 'value': '77.6' } },
  { 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2011' }, 'data':{ 'value': '79.4' } },
  { 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2012' }, 'data':{ 'value': '80.8' } },
  { 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2013' }, 'data':{ 'value': '86.6' } },
  { 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2014' }, 'data':{ 'value': '83.7' } },
  { 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2015' }, 'data':{ 'value': '84.5' } },
  { 'Labels':{ 'Xlabel': 'France', 'Ylabel': '2016' }, 'data':{ 'value': '82.6' } },
  { 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2010' }, 'data':{ 'value': '60.6' } },
  { 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2011' }, 'data':{ 'value': '65.4' } },
  { 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2012' }, 'data':{ 'value': '70.8' } },
  { 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2013' }, 'data':{ 'value': '73.8' } },
  { 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2014' }, 'data':{ 'value': '75.3' } },
  { 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2015' }, 'data':{ 'value': '77.5' } },
  { 'Labels':{ 'Xlabel': 'USA', 'Ylabel': '2016' }, 'data':{ 'value': '77.6' } },
  { 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2010' }, 'data':{ 'value': '64.9' } },
  { 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2011' }, 'data':{ 'value': '52.6' } },
  { 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2012' }, 'data':{ 'value': '60.8' } },
  { 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2013' }, 'data':{ 'value': '65.6' } },
  { 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2014' }, 'data':{ 'value': '52.6' } },
  { 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2015' }, 'data':{ 'value': '68.5' } },
  { 'Labels':{ 'Xlabel': 'Spain', 'Ylabel': '2016' }, 'data':{ 'value': '75.6' } },
  { 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2010' }, 'data':{ 'value': '55.6' } },
  { 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2011' }, 'data':{ 'value': '52.3' } },
  { 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2012' }, 'data':{ 'value': '54.8' } },
  { 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2013' }, 'data':{ 'value': '51.1' } },
  { 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2014' }, 'data':{ 'value': '55.6' } },
  { 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2015' }, 'data':{ 'value': '56.9' } },
  { 'Labels':{ 'Xlabel': 'China', 'Ylabel': '2016' }, 'data':{ 'value': '59.3' } },
  { 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2010' }, 'data':{ 'value': '43.6' } },
  { 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2011' }, 'data':{ 'value': '43.2' } },
  { 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2012' }, 'data':{ 'value': '55.8' } },
  { 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2013' }, 'data':{ 'value': '50.1' } },
  { 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2014' }, 'data':{ 'value': '48.5' } },
  { 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2015' }, 'data':{ 'value': '50.7' } },
  { 'Labels':{ 'Xlabel': 'Italy', 'Ylabel': '2016' }, 'data':{ 'value': '52.4' } },
  { 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2010' }, 'data':{ 'value': '28.2' } },
  { 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2011' }, 'data':{ 'value': '31.6' } },
  { 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2012' }, 'data':{ 'value': '29.8' } },
  { 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2013' }, 'data':{ 'value': '33.1' } },
  { 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2014' }, 'data':{ 'value': '32.6' } },
  { 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2015' }, 'data':{ 'value': '34.4' } },
  { 'Labels':{ 'Xlabel': 'UK', 'Ylabel': '2016' }, 'data':{ 'value': '35.8' } },
  { 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2010' }, 'data':{ 'value': '26.8' } },
  { 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2011' }, 'data':{ 'value': '29' } },
  { 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2012' }, 'data':{ 'value': '26.8' } },
  { 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2013' }, 'data':{ 'value': '27.6' } },
```

```

{ 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2014'}, 'data':{ 'value': '33' }},
{ 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2015'}, 'data':{ 'value': '35' }},
{ 'Labels':{ 'Xlabel': 'Germany', 'Ylabel': '2016'}, 'data':{ 'value': '35.6' }},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2010'}, 'data':{ 'value': '23.2' }},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2011'}, 'data':{ 'value': '24.9' }},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2012'}, 'data':{ 'value': '30.1' }},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2013'}, 'data':{ 'value': '22.2' }},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2014'}, 'data':{ 'value': '29.3' }},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2015'}, 'data':{ 'value': '32.1' }},
{ 'Labels':{ 'Xlabel': 'Mexico', 'Ylabel': '2016'}, 'data':{ 'value': '35' }},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2010'}, 'data':{ 'value':
'15.9' }},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2011'}, 'data':{ 'value':
'19.8' }},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2012'}, 'data':{ 'value':
'21.8' }},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2013'}, 'data':{ 'value':
'23.5' }},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2014'}, 'data':{ 'value':
'24.8' }},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2015'}, 'data':{ 'value':
'29.9' }},
{ 'Labels':{ 'Xlabel': 'Thailand', 'Ylabel': '2016'}, 'data':{ 'value':
'32.6' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2010'}, 'data':{ 'value': '22' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2011'}, 'data':{ 'value': '21.3' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2012'}, 'data':{ 'value': '24.2' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2013'}, 'data':{ 'value': '23.2' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2014'}, 'data':{ 'value': '25' }},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2015'}, 'data':{ 'value': '26.7'
}},
{ 'Labels':{ 'Xlabel': 'Austria', 'Ylabel': '2016'}, 'data':{ 'value': '28.1' }}
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Most Visited Destinations by International Tourist
Arrivals',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Austria', 'China', 'France', 'Germany', 'Italy',
'Mexico', 'Spain', 'Thailand', 'UK', 'USA'],
    },
    yAxis: {
        labels: ['2010', '2011', '2012', '2013', '2014', '2015',
'2016'],
    },
    dataSource: heatmapData,
    dataSourceSettings: {
        isJsonData: true,
        adaptorType: 'Cell',
        xDataMapping: 'Labels.Xlabel',

```

```

        yDataMapping: 'Labels.Ylabel',
        valueMapping: 'data.value'
    }, cellSettings: {
        border: {
            radius: 4,
            width: 1,
            color: 'white'
        },
        showLabel: true,
        format: '{value} M',
    }, paletteSettings: {
        palette: [{ color: '#DCD57E' },
        { color: '#A6DC7E' },
        { color: '#7EDCA2' },
        { color: '#6EB5D0' }
        ],
    }
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [To bind data for bubble heat map with size and color attributes](#)

Bubble heatmap in EJ2 JavaScript Heatmap chart control

Data points represent the data source values with **gradient** or **fixed** colors in the HeatMap. You can customize the appearance of these data points by changing the **color** and **size** attributes.

The data points can be represented in color fill or bubble shape by defining the [tileType](#) property.

By default, the data points are color filled with gradient or fixed colors and this depiction of data points is defined as **Rect** in the [tileType](#) property.

The cell customizations and color mapping for rect tile type is defined in [appearance](#) and [palette](#) sections in detail.

Bubble types

The data points can be represented in the bubble along with its attributes by setting the [tileType](#) property to **Bubble**.

In bubble HeatMap, you can display the data points with bubble size, bubble colors, and sector attributes of the bubble.

Bubble size

In this bubble HeatMap type, the size factor of the bubble is used to denote the data variations. The radius of the bubble varies according to data values.

By default, the bubble with small size denotes the data value with small magnitude and the larger bubble size denotes the data value with larger magnitude. This behavior can be inverted by using the [isInversedBubbleSize](#) property.

To render a bubble HeatMap with size series, set the [bubbleType](#) property to **Size**.

INDEX.TS

```
import { HeatMap, Legend, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
```

```

        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
        'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    paletteSettings: {
        palette: [
            { color: '#C06C84'},
            { color: '#6C5B7B'},
            { color: '#355C7D'}
        ],
        type: "Gradient"
    },
    cellSettings: {
        border: {
            width: 1
        },
        tileType: 'Bubble',
        bubbleType: 'Size',
        showLabel: false
    },
    dataSource: heatmapData,
    legendSettings: {
        visible: true,
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Bubble color

In HeatMap, defined with this tile type, the data points will be represented with same sized bubbles and the data value variations are represented with the bubble colors.

To represent the data points with variations in bubble colors, set the [bubbleType](#) property to **Color**.

INDEX.TS

```

import { HeatMap, Legend, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
        labelRotation: 45,
        labelIntersectAction: 'None'
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    paletteSettings: {
        palette: [
            { color: '#C06C84' },
            { color: '#6C5B7B' },
            { color: '#355C7D' }
        ],
        type: "Gradient"
    },
    cellSettings: {

```

```

        border: {
            width: 1
        },
        tileType: 'Bubble',
        bubbleType: 'Color'
    },
    dataSource: heatmapData,
    legendSettings: {
        visible: true,
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Bubble sector

In this bubble HeatMap type, the sector of the bubble decides the magnitude of data point. If the sector is large, then the data point value will be high.

To render the data points with bubble sector, set the [bubbleType](#) property to **Sector**.

INDEX.TS

```

import { HeatMap, Legend, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],

```



```

[99, 28, 22, 4, 66, 90],
[14, 26, 97, 69, 69, 3],
[7, 46, 47, 47, 88, 6],
[41, 55, 73, 23, 3, 79],
[56, 69, 21, 86, 3, 33],
[45, 7, 53, 81, 95, 79],
[60, 77, 74, 68, 88, 51],
[25, 25, 10, 12, 78, 14],
[25, 56, 55, 58, 12, 82],
[74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  paletteSettings: {
    palette: [
      { color: '#C06C84'},
      { color: '#6C5B7B'},
      { color: '#355C7D'}
    ],
    type: "Gradient"
  },
  cellSettings: {
    border: {
      width: 1
    },
    tileType: 'Bubble',
    bubbleType: 'Sector'
  },
  dataSource: heatmapData,
  legendSettings: {
    visible: true,
  }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Combining size and color bubble types

In this bubble HeatMap type, size and color of the bubble represents the data value variation. To render this bubble HeatMap type, set the [bubbleType](#) property to **SizeAndColor**.

The following examples demonstrate different data binding with the **SizeAndColor** bubble type set in the HeatMap.

<!-- markdownlint-disable MD036 -->

Array binding

When an array of numbers is specified as the data source, the bubble HeatMap can be rendered with different sizes and colors depending on the bound data.

<!-- markdownlint-disable MD036 -->

Table

The following example illustrates how to render a bubble HeatMap with different sizes and colors using array table binding.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, BubbleTooltipData, ITooltipEventArgs }
from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip);
let heatmapData: any [] = [
    [[4,39], [3,8], [1,3], [1,10], [4,4], [2,15]],
    [[4,28], [5,92], [5,73], [3,1], [3,4], [4,126]],
    [[4,45], [5,152], [0,44], [4,54], [5,243], [2,45]]
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {

```

```

        text: 'Commercial Aviation Accidents and Fatalities by year 2012
- 2017',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        },
    },
    xAxis: {
        labels: ['2017', '2016', '2015']
    },
    yAxis: {
        labels: ['Jan-Feb', 'Mar-Apr', 'May-Jun', 'Jul-Aug', 'Sep-Oct',
'Nov-Dec']
    },
    paletteSettings: {
        palette: [
            { color: '#C06C84' },
            { color: '#6C5B7B' },
            { color: '#355C7D' }
        ],
        type: "Gradient"
    },
    cellSettings: {
        border: {
            width: 1
        },
        tileType: 'Bubble',
        bubbleType: 'SizeAndColor'
    },
    dataSource: heatmapData,
    legendSettings: {
        visible: true,
    },
    tooltipRender: (args: ITooltipEventArgs) => {
        args.content = ['Year ' + ' : ' + args.xLabel + '<br/>' + 'Months '
+ ' : ' + args.yLabel + '<br/>'
            + 'Accidents ' + ' : ' + (args.value as
BubbleTooltipData[])[0].bubbleData + '<br/>' + 'Fatalities ' + ' : '
            + (args.value as BubbleTooltipData[])[1].bubbleData];
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Cell

The following example illustrates how to render a bubble HeatMap with different sizes and colors using array cell binding.

INDEX.TS

```

import { HeatMap, Legend, Adaptor, Tooltip, BubbleTooltipData,
ITooltipEventArgs } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Adaptor, Tooltip);
let heatmapData: any [] = [
    [0,0,[4,39]], [0,1,[3,8]], [0,2,[1,3]], [0,3,[1,10]], [0,4,[4,4]],
    [0,5,[2,15]],
    [1,0,[4,28]], [1,1,[5,92]], [1,2,[5,73]], [1,3,[3,1]], [1,4,[3,4]],
    [1,5,[4,126]],
    [2,0,[4,45]], [2,1,[5,152]], [2,2,[0,44]], [2,3,[4,54]], [2,4,[5,243]],
    [2,5,[2,45]]
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Commercial Aviation Accidents and Fatalities by year 2012
- 2017',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['2017', '2016', '2015']
    },
    yAxis: {
        labels: ['Jan-Feb', 'Mar-Apr', 'May-Jun', 'Jul-Aug', 'Sep-Oct',
'Nov-Dec']
    },

```

```

        paletteSettings: {
            palette: [
                { color: '#C06C84'},
                { color: '#6C5B7B'},
                { color: '#355C7D'}
            ],
            type: "Gradient"
        },
        cellSettings: {
            border: {
                width: 1
            },
            tileType: 'Bubble',
            bubbleType: 'SizeAndColor'
        },
        dataSource: heatmapData,
        dataSourceSettings: {
            isJsonData: false,
            adaptorType: 'Cell'
        },
        legendSettings: {
            visible: true,
        },
        tooltipRender: (args: ITooltipEventArgs) => {
            args.content = ['Year ' + ' : ' + args.xLabel + '<br/>' + 'Months ' + ' : ' + args.yLabel + '<br/>'
                + 'Accidents ' + ' : ' + (args.value as
BubbleTooltipData[])[0].bubbleData + '<br/>' + 'Fatalities ' + ' : '
                + (args.value as BubbleTooltipData[])[1].bubbleData];
        }
    });
    heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

JSON binding

When a list of JSON objects are specified as data source, the bubble HeatMap can be rendered with different sizes and colors depending on the bound data.

<!-- markdownlint-disable MD036 -->

Table

The following example illustrates how to render a bubble HeatMap with different sizes and colors using JSON table binding.

INDEX.TS

```

import { HeatMap, Legend, Adaptor, Tooltip, BubbleTooltipData,
ITooltipEventArgs } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Adaptor, Tooltip);
let heatmapData: object = [
    { 'Year': '2017', 'Jan-Feb': [4,39], 'Mar-Apr': [3,8], 'May-Jun':
[1,3], 'Jul-Aug': [1,10], 'Sep-Oct': [4,4], 'Nov-Dec': [2,15]},
    { 'Year': '2016', 'Jan-Feb': [4,28], 'Mar-Apr': [5,92], 'May-Jun':
[5,73], 'Jul-Aug': [3,1], 'Sep-Oct': [3,4], 'Nov-Dec': [4,126]},
    { 'Year': '2015', 'Jan-Feb': [4,45], 'Mar-Apr': [5,152], 'May-Jun':
[0,44], 'Jul-Aug': [4,54], 'Sep-Oct': [5,243], 'Nov-Dec': [2,45]};
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Commercial Aviation Accidents and Fatalities by year 2012
- 2017',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['2017', '2016', '2015']
    },
    yAxis: {
        labels: ['Jan-Feb', 'Mar-Apr', 'May-Jun', 'Jul-Aug', 'Sep-Oct',
'Nov-Dec']
    },
    paletteSettings: {
        palette: [
            { color: '#C06C84'},
            { color: '#6C5B7B'},
            { color: '#355C7D'}
        ]
    }
});

```

```

        type: "Gradient"
    },
    cellSettings: {
        border: {
            width: 1
        },
        tileType: 'Bubble',
        bubbleType: 'SizeAndColor'
    },
    dataSource: heatmapData,
    dataSourceSettings: {
        isJsonData: true,
        adaptorType: 'Table',
        xDataMapping: 'Year',
    },
    legendSettings: {
        visible: true,
    },
    tooltipRender: (args: ITooltipEventArgs) => {
        args.content = ['Year ' + ' : ' + args.xLabel + '<br/>' + 'Months '
+ ' : ' + args.yLabel + '<br/>'
            + 'Accidents ' + ' : ' + (args.value as
BubbleTooltipData[])[0].bubbleData + '<br/>' + 'Fatalities ' + ' : '
            + (args.value as BubbleTooltipData[])[1].bubbleData];
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

```
<!-- markdownlint-disable MD036 -->
```

Cell

The following example illustrates how to render a bubble HeatMap with different sizes and colors using JSON cell binding.

INDEX.TS

```
import { HeatMap, Legend, Adaptor, Tooltip, BubbleTooltipData,
ITooltipEventArgs } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Adaptor, Tooltip);
let jsonCellBubbleData: Object = [
    { Year: '2017', Months: 'Jan-Feb', Accidents: 4, Fatalities: 39 },
    { Year: '2017', Months: 'Mar-Apr', Accidents: 3, Fatalities: 8 },
    { Year: '2017', Months: 'May-Jun', Accidents: 1, Fatalities: 3 },
    { Year: '2017', Months: 'Jul-Aug', Accidents: 1, Fatalities: 10 },
    { Year: '2017', Months: 'Sep-Oct', Accidents: 4, Fatalities: 4 },
    { Year: '2017', Months: 'Nov-Dec', Accidents: 2, Fatalities: 15 },
    { Year: '2016', Months: 'Jan-Feb', Accidents: 4, Fatalities: 28 },
    { Year: '2016', Months: 'Mar-Apr', Accidents: 5, Fatalities: 92 },
    { Year: '2016', Months: 'May-Jun', Accidents: 5, Fatalities: 73 },
    { Year: '2016', Months: 'Jul-Aug', Accidents: 3, Fatalities: 1 },
    { Year: '2016', Months: 'Sep-Oct', Accidents: 3, Fatalities: 4 },
    { Year: '2016', Months: 'Nov-Dec', Accidents: 4, Fatalities: 126 },
    { Year: '2015', Months: 'Jan-Feb', Accidents: 4, Fatalities: 45 },
    { Year: '2015', Months: 'Mar-Apr', Accidents: 5, Fatalities: 152 },
    { Year: '2015', Months: 'May-Jun', Accidents: 0, Fatalities: 0 },
    { Year: '2015', Months: 'Jul-Aug', Accidents: 4, Fatalities: 54 },
    { Year: '2015', Months: 'Sep-Oct', Accidents: 5, Fatalities: 243 },
    { Year: '2015', Months: 'Nov-Dec', Accidents: 2, Fatalities: 45 },
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Commercial Aviation Accidents and Fatalities by year 2012
- 2017',
        textStyle: {
```



```

        size: '15px',
        fontWeight: '500',
        fontStyle: 'Normal',
        fontFamily: 'Segoe UI'
    },
    },
    xAxis: {
        labels: ['2017', '2016', '2015']
    },
    yAxis: {
        labels: ['Jan-Feb', 'Mar-Apr', 'May-Jun', 'Jul-Aug', 'Sep-Oct',
'Nov-Dec']
    },
    paletteSettings: {
        palette: [
            { color: '#C06C84'},
            { color: '#6C5B7B'},
            { color: '#355C7D'}
        ],
        type: "Gradient"
    },
    cellSettings: {
        border: {
            width: 1
        },
        tileType: 'Bubble',
        bubbleType: 'SizeAndColor'
    },
    dataSource: jsonCellBubbleData,
    dataSourceSettings: {
        isJsonData: true,
        adaptorType: 'Cell',
        xDataMapping: 'Year',
        yDataMapping: 'Months',
        bubbleDataMapping: { size: 'Accidents', color: 'Fatalities' }
    },
    legendSettings: {
        visible: true,
    },
    tooltipRender: (args: ITooltipEventArgs) => {
        args.content = ['Year ' + ' : ' + args.xLabel + '<br/>' + 'Months '
+ ' : ' + args.yLabel + '<br/>'
            + 'Accidents ' + ' : ' + (args.value as
BubbleTooltipData[])[0].bubbleData + '<br/>' + 'Fatalities ' + ' : '
            + (args.value as BubbleTooltipData[])[1].bubbleData];
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Binding size and color values from datasource

The size and color of the bubbles in the **SizeAndColor** bubble HeatMap type can be customized by binding the datasource field name that holds the size and color values to the [size](#) and [color](#) properties in the [bubbleDataMapping](#).

The `bubbleDataMapping` supports only for the JSON data with cell adaptor type.

INDEX.TS

```

import { HeatMap, Legend, Adaptor, Tooltip, BubbleTooltipData,
ITooltipEventArgs } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Adaptor, Tooltip);
let jsonCellBubbleData: Object = [
    { Year: '2017', Months: 'Jan-Feb', Accidents: 4, Fatalities: 39 },
    { Year: '2017', Months: 'Mar-Apr', Accidents: 3, Fatalities: 8 },
    { Year: '2017', Months: 'May-Jun', Accidents: 1, Fatalities: 3 },
    { Year: '2017', Months: 'Jul-Aug', Accidents: 1, Fatalities: 10 },
    { Year: '2017', Months: 'Sep-Oct', Accidents: 4, Fatalities: 4 },
    { Year: '2017', Months: 'Nov-Dec', Accidents: 2, Fatalities: 15 },
    { Year: '2016', Months: 'Jan-Feb', Accidents: 4, Fatalities: 28 },
    { Year: '2016', Months: 'Mar-Apr', Accidents: 5, Fatalities: 92 },
    { Year: '2016', Months: 'May-Jun', Accidents: 5, Fatalities: 73 },
],

```

```

        { Year: '2016', Months: 'Jul-Aug', Accidents: 3, Fatalities: 1
    },
        { Year: '2016', Months: 'Sep-Oct', Accidents: 3, Fatalities: 4
    },
        { Year: '2016', Months: 'Nov-Dec', Accidents: 4, Fatalities: 126
    },
        { Year: '2015', Months: 'Jan-Feb', Accidents: 4, Fatalities: 45
    },
        { Year: '2015', Months: 'Mar-Apr', Accidents: 5, Fatalities: 152
    },
        { Year: '2015', Months: 'May-Jun', Accidents: 0, Fatalities: 0
    },
        { Year: '2015', Months: 'Jul-Aug', Accidents: 4, Fatalities: 54
    },
        { Year: '2015', Months: 'Sep-Oct', Accidents: 5, Fatalities: 243
    },
        { Year: '2015', Months: 'Nov-Dec', Accidents: 2, Fatalities: 45
    },
    ];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Commercial Aviation Accidents and Fatalities by year 2012
- 2017',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['2017', '2016', '2015']
    },
    yAxis: {
        labels: ['Jan-Feb', 'Mar-Apr', 'May-Jun', 'Jul-Aug', 'Sep-Oct',
'Nov-Dec']
    },
    paletteSettings: {
        palette: [
            { color: '#C06C84'},
            { color: '#6C5B7B'},
            { color: '#355C7D'}
        ],
        type: "Gradient"
    },
    cellSettings: {
        border: {
            width: 1
        },
        tileType: 'Bubble',
        bubbleType: 'SizeAndColor'
    },
    dataSource: jsonCellBubbleData,
    dataSourceSettings: {
        isJsonData: true,
        adaptorType: 'Cell',
        xDataMapping: 'Year',

```

```

        yDataMapping: 'Months',
        bubbleDataMapping: { size: 'Accidents', color: 'Fatalities' }
    },
    legendSettings: {
        visible: true,
    },
    tooltipRender: (args: ITooltipEventArgs) => {
        args.content = ['Year ' + ' : ' + args.xLabel + '<br/>' + 'Months '
+ ' : ' + args.yLabel + '<br/>'
        + 'Accidents ' + ' : ' + (args.value as
BubbleTooltipData[])[0].bubbleData + '<br/>' + 'Fatalities ' + ' : '
        + (args.value as BubbleTooltipData[])[1].bubbleData];
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Rendering mode in EJ2 JavaScript Heatmap chart control

Heat map can be displayed using **Canvas** or **Scalable Vector Graphics (SVG)** rendering logic to improve the initial load performance and scalability. Heat map can also be automatically switched between **Canvas** and **SVG** modes based on dataset size. You can enable this mode by

setting the [renderingMode](#) property as **Auto**.

If the **Auto** mode is enabled in the heat map and there are more than 10,000 data points, then the heat map will be rendered in a **Canvas** mode; Otherwise, the heat map will be rendered in a **SVG** mode.

INDEX.TS

```
import { HeatMap, Legend, Tooltip, Adaptor } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    cellSettings: {
        showLabel: true,
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    renderingMode: "SVG",
    dataSource: heatmapData,
    showTooltip: true,
});
heatmap.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis in EJ2 JavaScript HeatMap chart control

HeatMap consists of two axes namely, X-axis and Y-axis that displays the row headers and column headers to plot the data points respectively.

You can define the type, format, and other customizing options for both axes in the HeatMap.

Types

There are three different axis types available in the HeatMap, which defines the data type of the axis labels. You can define the axis type by using the [valueType](#) property in the HeatMap.

Category axis

Category axis type is used to represent the string values in axis labels.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, Adaptor } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    xAxis: {
        valueType: "Category",
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin', 'Mario'],
    },
    yAxis: {

```

```

        valueType: "Category",
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Numeric axis

Numeric axis type is used to represent the numeric values in axis labels.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, Adaptor } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],

```

```

[74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  xAxis: {
    valueType: "Numeric",
    minimum: 0,
    maximum: 11,
  },
  yAxis: {
    valueType: "Numeric",
    minimum: 0,
    maximum: 5,
  },
  dataSource: heatmapData
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Date-time axis

Date-time axis type is used to represent the date-time values in axis labels with a specific format. In date-time axis, you can define the start and end date/time using the [minimum](#) and [maximum](#) properties.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, Adaptor} from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);

```



```

let heatmapData: any [] = [
    [36371, 25675, 28292, 33399, 35980, 38585, 39351, 39964, 36543,
    30529, 33298, 36985],
    [34702, 27618, 31063, 34525, 36772, 35410, 38750, 39467, 35390,
    34196, 35302, 35703],
    [34522, 31324, 32128, 34231, 36817, 34381, 37180, 38255, 32776,
    32645, 31539, 32981],
    [32213, 28755, 29517, 31214, 33747, 33507, 35763, 36837, 32910,
    33437, 30659, 31965],
    [31282, 28663, 32952, 33941, 34506, 36875, 38836, 35497, 34285,
    34094, 32256, 33699],
    [31714, 29405, 33745, 32838, 33461, 35034, 36122, 37943, 34128,
    30624, 32398, 33522],
    [32064, 28387, 33751, 32537, 34034, 35977, 37196, 38301, 33627,
    34115, 31072, 33939],
    [32417, 27868, 30807, 33386, 35284, 36126, 39753, 40978, 35777,
    35277, 31281, 35411],
    [32494, 29848, 34385, 35804, 37943, 38722, 41315, 41335, 37177,
    37443, 32457, 37304],
    [34378, 29576, 30547, 35664, 36622, 38145, 40347, 41868, 38252,
    36505, 29576, 36450],
    [35219, 31670, 32589, 34927, 36998, 39825, 41126, 42002, 37021,
    36583, 32408, 37108]
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Monthly Flight Traffic at JFK Airport',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        valueType: "DateTime",
        minimum: new Date(2007, 0, 1),
        maximum: new Date(2017, 0, 1),
        intervalType: "Years",
        labelFormat: "yyyy",
        labelRotation: 45,
        labelIntersectAction: 'None'
    },
    yAxis: {
        labels: ['Jan', 'Feb', 'Mar', 'Apr', 'May',
            'Jun', 'July', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec'],
    },
    legendSettings: {
        visible: false,
    },
    cellSettings: {
        showLabel: false,
        border: {
            width: 0,
        },
        format: '{value} flights'
    },
});

```

```

        dataSource: heatmapData
    });
    heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Inversed axis

HeatMap supports inverting the axis origin for both axes, where the axis labels are placed in an inversed manner. You can enable axis inverting by enabling the [isInversed](#) property.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, Adaptor } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: any [] = [
  [36371, 25675, 28292, 33399, 35980, 38585, 39351, 39964, 36543,
  30529, 33298, 36985],
  [34702, 27618, 31063, 34525, 36772, 35410, 38750, 39467, 35390,
  34196, 35302, 35703],
  [34522, 31324, 32128, 34231, 36817, 34381, 37180, 38255, 32776,
  32645, 31539, 32981],
  [32213, 28755, 29517, 31214, 33747, 33507, 35763, 36837, 32910,
  33437, 30659, 31965],
  [31282, 28663, 32952, 33941, 34506, 36875, 38836, 35497, 34285,
  34094, 32256, 33699],
  [31714, 29405, 33745, 32838, 33461, 35034, 36122, 37943, 34128,
  30624, 32398, 33522],

```

```

[32064, 28387, 33751, 32537, 34034, 35977, 37196, 38301, 33627,
34115, 31072, 33939],
[32417, 27868, 30807, 33386, 35284, 36126, 39753, 40978, 35777,
35277, 31281, 35411],
[32494, 29848, 34385, 35804, 37943, 38722, 41315, 41335, 37177,
37443, 32457, 37304],
[34378, 29576, 30547, 35664, 36622, 38145, 40347, 41868, 38252,
36505, 29576, 36450],
[35219, 31670, 32589, 34927, 36998, 39825, 41126, 42002, 37021,
36583, 32408, 37108]
];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Monthly Flight Traffic at JFK Airport',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    valueType: "DateTime",
    minimum: new Date(2007, 0, 1),
    maximum: new Date(2017, 0, 1),
    intervalType: "Years",
    labelFormat: "yyyy",
    labelRotation: 45,
    labelIntersectAction: 'None',
    isInversed: true
  },
  yAxis: {
    labels: ['Jan', 'Feb', 'Mar', 'Apr', 'May',
      'Jun', 'July', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec'],
  },
  legendSettings: {
    visible: false,
  },
  cellSettings: {
    showLabel: false,
    border: {
      width: 0,
    },
    format: '{value} flights'
  },
  dataSource: heatmapData
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Opposed axis

In HeatMap, you can place the axis label in an opposite position of its default axis label position by using the [opposedPosition](#) property.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, Adaptor } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: any [] = [
    [36371, 25675, 28292, 33399, 35980, 38585, 39351, 39964, 36543,
30529, 33298, 36985],
    [34702, 27618, 31063, 34525, 36772, 35410, 38750, 39467, 35390,
34196, 35302, 35703],
    [34522, 31324, 32128, 34231, 36817, 34381, 37180, 38255, 32776,
32645, 31539, 32981],
    [32213, 28755, 29517, 31214, 33747, 33507, 35763, 36837, 32910,
33437, 30659, 31965],
    [31282, 28663, 32952, 33941, 34506, 36875, 38836, 35497, 34285,
34094, 32256, 33699],
    [31714, 29405, 33745, 32838, 33461, 35034, 36122, 37943, 34128,
30624, 32398, 33522],
    [32064, 28387, 33751, 32537, 34034, 35977, 37196, 38301, 33627,
34115, 31072, 33939],
    [32417, 27868, 30807, 33386, 35284, 36126, 39753, 40978, 35777,
35277, 31281, 35411],
    [32494, 29848, 34385, 35804, 37943, 38722, 41315, 41335, 37177,
37443, 32457, 37304],
    [34378, 29576, 30547, 35664, 36622, 38145, 40347, 41868, 38252,
36505, 29576, 36450],
    [35219, 31670, 32589, 34927, 36998, 39825, 41126, 42002, 37021,
36583, 32408, 37108]
];
let heatmap: HeatMap = new HeatMap({

```

```

    titleSettings: {
      text: 'Monthly Flight Traffic at JFK Airport',
      textStyle: {
        size: '15px',
        fontWeight: '500',
        fontStyle: 'Normal',
        fontFamily: 'Segoe UI'
      }
    },
    xAxis: {
      valueType: "DateTime",
      minimum: new Date(2007, 0, 1),
      maximum: new Date(2017, 0, 1),
      intervalType: "Years",
      labelFormat: "yyyy",
      labelRotation: 45,
      labelIntersectAction: 'None',
      opposedPosition: true
    },
    yAxis: {
      labels: ['Jan', 'Feb', 'Mar', 'Apr', 'May',
        'Jun', 'July', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec'],
    },
    legendSettings: {
      visible: false,
    },
    cellSettings: {
      showLabel: false,
      border: {
        width: 0,
      },
      format: '{value} flights'
    },
    dataSource: heatmapData
  });
  heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis labels customization

Customizing the text style

The text style of the axis labels can be customized using the following options available in the [textStyle](#) property.

- [color](#) - It is used to change the text color of the axis labels.
- [fontFamily](#) - It is used to change the font family of the axis labels.
- [fontStyle](#) - It is used to change the font style of the axis labels.
- [fontWeight](#) - It is used to change the font weight of the axis labels.
- [size](#) - It is used to change the font size of the axis labels.
- [textAlignment](#) - It is used to position and align the axis labels. This property allows you to specify values such as **Near**, **Center**, and **Far**.
- [textOverflow](#) - When the axis label exceeds the intended space, this property is used to trim or wrap it. This property takes values such as **None**, **Trim**, and **Wrap**.

INDEX.TS

```

import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [52, 65, 67, 45, 37, 52, 32],
    [68, 52, 63, 51, 30, 51, 51],
    [60, 50, 42, 53, 66, 70, 41],
    [66, 64, 46, 40, 47, 41, 45],
    [65, 42, 58, 32, 36, 44, 49],
    [54, 46, 61, 46, 40, 39, 41],
    [48, 46, 61, 47, 49, 41, 41],
    [69, 52, 41, 44, 41, 52, 46],
    [50, 59, 44, 43, 27, 42, 26],
    [47, 49, 66, 53, 50, 34, 31],
    [61, 40, 62, 26, 34, 54, 56]
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Product wise Monthly sales revenue for a e-commerce website',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI',
        },
    },
});

```

```

xAxis: {
  textStyle: {
    color: 'red',
    size: '15px',
    fontWeight: '650',
    fontStyle: 'Normal',
    fontFamily: 'Segoe UI',
    textAlign: 'Center',
    textOverflow: 'Wrap'
  },
  opposedPosition: true,
  labels: [
    'Month of Feburary 2023',
    'Month of March 2023',
    'Month of April 2023',
    'Month of May 2023',
    'Month of June 2023',
    'Month of July 2023',
    'Month of August 2023',
    'Month of September 2023',
    'Month of October 2023',
    'Month of November 2023',
    'Month of December 2023'
  ],
},
yAxis: {
  textStyle: {
    color: 'red',
    size: '15px',
    fontWeight: '650',
    fontStyle: 'Normal',
    fontFamily: 'Segoe UI',
    textAlign: 'Center',
    textOverflow: 'Wrap',
  },
  maxLabelLength: 70,
  labels: [
    'Ace Apparels',
    'Alpha Apparels',
    'RL Garments',
    'RRD Garments',
    'RRD Apparels',
    'RR Garments',
    'SR Garments',
  ],
},
legendSettings: {
  visible: false,
},
paletteSettings: {
  palette: [{ color: '#F0C27B' }, { color: '#4B1248' }],
},
dataSource: heatmapData
}); heatmap.appendTo('#element');

```

[INDEX.HTML](#)

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Providing line breaks

Axis labels with line breaks improve the readability of the HeatMap by splitting the text on an axis into multiple lines. The “`
`” character is used to add line breaks to the axis labels.

INDEX.TS

```

import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any[] = [
  [1, 76],
  [19, 3]
];
let heatmap: HeatMap = new HeatMap({
  xAxis: {
    labels: ['Actual<br>Accept', 'Actual<br>Reject'],
    opposedPosition: true,
  },
  yAxis: {
    labels: ['Actual<br>Accept', 'Actual<br>Reject'],
    maxLabelLength: 50,
  },
  dataSource: heatmapData
});
heatmap.appendTo('#element');

```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing labels when intersecting with other labels

When the axis labels intersect, [labelIntersectAction](#) property is used to handle the intersection. The [labelIntersectAction](#) property can take the following values.

- **None** - It specifies that no action is taken when the axis labels intersect.
- **Trim** - It specifies to trim the axis labels when they intersect.
- **Rotate45** - When the axis labels intersect, they are rotated to 45 degrees.
- **MultipleRows** - It specifies to show all the axis labels as multiple rows when they intersect.

The below example demonstrates to trim the axis labels by using the [labelIntersectAction](#) property.

INDEX.TS

```

import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
  [52, 65, 67, 45, 37, 52, 32],
  [68, 52, 63, 51, 30, 51, 51],
  [60, 50, 42, 53, 66, 70, 41],
  [66, 64, 46, 40, 47, 41, 45],
  [65, 42, 58, 32, 36, 44, 49],
  [54, 46, 61, 46, 40, 39, 41],
  [48, 46, 61, 47, 49, 41, 41],
  [69, 52, 41, 44, 41, 52, 46],
  [50, 59, 44, 43, 27, 42, 26],
  [47, 49, 66, 53, 50, 34, 31],

```

```

[61, 40, 62, 26, 34, 54, 56]
];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Product wise Monthly sales revenue for a e-commerce website',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI',
    },
  },
  xAxis: {
    enableTrim: true,
    labelIntersectAction: 'Trim',
    opposedPosition: true,
    labels: [
      'Month of Feburary 2023',
      'Month of March 2023',
      'Month of April 2023',
      'Month of May 2023',
      'Month of June 2023',
      'Month of July 2023',
      'Month of August 2023',
      'Month of September 2023',
      'Month of October 2023',
      'Month of November 2023',
      'Month of December 2023'
    ],
  },
  yAxis: {
    enableTrim: true,
    labelIntersectAction: 'Trim',
    labels: [
      'Ace Apparels',
      'Alpha Apparels',
      'RL Garments',
      'RRD Garments',
      'RRD Apparels',
      'RR Garments',
      'SR Garments',
    ],
  },
  legendSettings: {
    visible: false,
  },
  paletteSettings: {
    palette: [{ color: '#F0C27B' }, { color: '#4B1248' }],
  },
  dataSource: heatmapData
}); heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Rotating labels

The axis labels can be rotated to the desired angles by using the [labelRotation](#) property.

INDEX.TS

```

import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [52, 65, 67, 45, 37, 52, 32],
    [68, 52, 63, 51, 30, 51, 51],
    [60, 50, 42, 53, 66, 70, 41],
    [66, 64, 46, 40, 47, 41, 45],
    [65, 42, 58, 32, 36, 44, 49],
    [54, 46, 61, 46, 40, 39, 41],
    [48, 46, 61, 47, 49, 41, 41],
    [69, 52, 41, 44, 41, 52, 46],
    [50, 59, 44, 43, 27, 42, 26],
    [47, 49, 66, 53, 50, 34, 31],
    [61, 40, 62, 26, 34, 54, 56]
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Product wise Monthly sales revenue for a e-commerce website',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI',
        },
    },
},

```

```

xAxis: {
  labelRotation: 90,
  opposedPosition: true,
  labels: [
    'Month of Feburary 2023',
    'Month of March 2023',
    'Month of April 2023',
    'Month of May 2023',
    'Month of June 2023',
    'Month of July 2023',
    'Month of August 2023',
    'Month of September 2023',
    'Month of October 2023',
    'Month of November 2023',
    'Month of December 2023'
  ],
},
yAxis: {
  labelRotation: 45,
  labels: [
    'Ace Apparels',
    'Alpha Apparels',
    'RL Garments',
    'RRD Garments',
    'RRD Apparels',
    'RR Garments',
    'SR Garments'
  ],
},
legendSettings: {
  visible: false,
},
paletteSettings: {
  palette: [{ color: '#F0C27B' }, { color: '#4B1248' }],
},
dataSource: heatmapData
}); heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Label formatting

HeatMap supports formatting the axis labels by using the [labelFormat](#) property. Using this property, you can customize the axis label by global string format ('P', 'C', etc) or customized format like '{value}°C'.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, Adaptor } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]
];
let heatmap: HeatMap = new HeatMap({
  xAxis: {
    valueType: "DateTime",
    minimum: new Date(2007, 0, 1),
    intervalType: "Years",
    labelFormat: "yyyy",
  },
  yAxis: {
    valueType: "Numeric",
    labelFormat: "${value}"
  },
  legendSettings: {
    visible: false,
  },
  dataSource: heatmapData
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 HeatMap</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis intervals

In HeatMap, you can define an interval between the axis labels using the [interval](#) property. In date-time axis, you can change the interval mode by using the [intervalType](#) property. The date-time axis supports the following interval types:

- Years
- Months
- Days
- Hours
- Minutes

INDEX.TS

```

import { HeatMap, Legend, Tooltip, Adaptor } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip, Adaptor);
let heatmapData: any [] = [
    [36371, 25675, 28292, 33399, 35980, 38585, 39351, 39964, 36543,
    30529, 33298, 36985],
    [34702, 27618, 31063, 34525, 36772, 35410, 38750, 39467, 35390,
    34196, 35302, 35703],
    [34522, 31324, 32128, 34231, 36817, 34381, 37180, 38255, 32776,
    32645, 31539, 32981],
    [32213, 28755, 29517, 31214, 33747, 33507, 35763, 36837, 32910,
    33437, 30659, 31965],
    [31282, 28663, 32952, 33941, 34506, 36875, 38836, 35497, 34285,
    34094, 32256, 33699],

```

```

        [31714, 29405, 33745, 32838, 33461, 35034, 36122, 37943, 34128,
        30624, 32398, 33522],
        [32064, 28387, 33751, 32537, 34034, 35977, 37196, 38301, 33627,
        34115, 31072, 33939],
        [32417, 27868, 30807, 33386, 35284, 36126, 39753, 40978, 35777,
        35277, 31281, 35411],
        [32494, 29848, 34385, 35804, 37943, 38722, 41315, 41335, 37177,
        37443, 32457, 37304],
        [34378, 29576, 30547, 35664, 36622, 38145, 40347, 41868, 38252,
        36505, 29576, 36450],
        [35219, 31670, 32589, 34927, 36998, 39825, 41126, 42002, 37021,
        36583, 32408, 37108]
    ];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Monthly Flight Traffic at JFK Airport',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        valueType: "DateTime",
        minimum: new Date(2007, 0, 1),
        maximum: new Date(2017, 0, 1),
        intervalType: "Years",
        interval: 2,
        labelFormat: "yyyy"
    },
    yAxis: {
        labels: ['Jan', 'Feb', 'Mar', 'Apr', 'May',
            'Jun', 'July', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec'],
    },
    legendSettings: {
        visible: false,
    },
    cellSettings: {
        showLabel: false,
        border: {
            width: 0,
        },
        format: '{value} flights'
    },
    dataSource: heatmapData
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis label increment

Axis label increment in the HeatMap is used to display the axis labels with regular interval values in numeric and date-time axes. The labels will be displayed with tick gaps when you set the label interval. But, to achieve the same behavior without tick gaps, use the label increment. You can set the axis label increment using the [increment](#) property and the default value of this property is **1**.

INDEX.TS

```

import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    xAxis: {
        valueType: "Numeric",
        minimum: 0,
        increment: 2
    },
    yAxis: {
        valueType: "Numeric",
        minimum: 0,
        increment: 3
    }
});

```



```

    },
    dataSource: heatmapData
  });
  heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Limiting labels in date-time axis

You can display the axis labels at specific time intervals along with the date-time axis using the [showLabelOn](#) property. This property supports the following types:

- None: Displays the axis labels based on the `intervalType` and `interval` property of the axis. This type is default value of the `showLabelOn` property.
- Years: Displays the axis labels on every year between given date-time range.
- Months: Displays the axis labels on every month between given date-time range.
- Days: Displays the axis labels on every day between given date-time range.
- Minutes: Displays the axis labels on every minute between given date-time range.

INDEX.TS

```

import { HeatMap, Legend, Tooltip, ITooltipEventArgs } from '@syncfusion/ej2-heatmap';
import { Internationalization } from '@syncfusion/ej2-base';
HeatMap.Inject(Legend, Tooltip);
let heatmapData: any [] = [

```

```

[null, null, null, null, 16, 48, 0],
[0, 15, 0, 24, 0, 39, 0],
[0, 18, 37, 0, 0, 50, 0],
[0, 10, 0, 0, 44, 5, 0],
[0, 36, 0, 45, 20, 18, 0],
[0, 28, 1, 42, 0, 10, 0],
[0, 16, 32, 0, 1, 25, 0],
[0, 31, 2, 9, 24, 0, 0],
[0, 8, 47, 0, 0, 35, 0],
[0, 31, 0, 0, 0, 40, 0],
[0, 8, 0, 27, 0, 35, 0],
[0, 12, 9, 45, 0, 8, 0],
[0, 0, 13, 0, 22, 10, 0],
[0, 16, 32, 0, 1, 25, 0],
[0, 31, 2, 9, 24, 0, 0],
[0, 8, 47, 27, 0, 35, 0],
[0, 28, 14, 10, 0, 0, 0],
[0, 36, 0, 45, 20, 18, 0],
[0, 28, 1, 42, 0, 10, 0],
[0, 31, 0, 24, 0, 40, 0],
[0, 8, 47, 27, 0, 35, 0],
[0, 36, 0, 45, 20, 18, 0],
[0, 28, 1, 42, 0, 10, 0],
[0, 31, 0, 24, 0, 40, 0],
[0, 16, 32, 0, 1, 25, 0],
[0, 31, 2, 9, 24, 0, 0],
[0, 8, 47, 27, 0, 35, 0],
[0, 10, 0, 36, 23, 19, 0],
[0, 18, 37, 23, 0, 50, 0],
[0, 28, 14, 10, 0, 0, 0],
[0, 18, 37, 23, 0, 50, 0],
[0, 18, 37, 23, 0, 50, 0],
[0, 28, 14, 10, 0, 0, 0],
[0, 31, 2, 9, 24, 0, 0],
[0, 8, 47, 27, 0, 35, 0],
[0, 10, 2, 0, 44, 5, 0],
[0, 36, 0, 45, 20, 18, 0],
[0, 28, 1, 42, 0, 10, 0],
[0, 31, 0, 24, 0, 40, 1],
[0, 16, 32, 0, 1, 25, 0],
[0, 31, 2, 9, 24, 0, 0],
[0, 8, 47, 27, 0, 35, 0],
[0, 10, 2, 0, 44, 5, 0],
[0, 12, 9, 45, 0, 8, 0],
[0, 0, 13, 35, 22, 10, 0],
[0, 28, 14, 10, 0, 0, 0],
[0, 36, 0, 45, 20, 18, 2],
[0, 28, 1, 42, 0, 10, 0],
[0, 31, 0, 24, 0, 40, 1],
[0, 8, 47, 27, 0, 35, 0],
[0, 10, 2, 0, 44, 5, 0],
[0, 31, 2, 9, 24, 0, 1],
[0, 8, 47, 27, 0, 35, 40],
[0, 10, 2, 0, 44, 5, null]
];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {

```

```

        text: 'Annual Summary of User Activities in GitLab',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        },
    },
    height: '280px',
    xAxis: {
        opposedPosition: true,
        valueType: 'DateTime',
        minimum: new Date(2017, 6, 23),
        maximum: new Date(2018, 6, 30),
        intervalType: 'Days',
        showLabelOn: 'Months',
        labelFormat: 'MMM',
        increment: 7,
    },
    yAxis: {
        labels: ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat'],
        isInversed: true,
    },
    cellSettings: {
        showLabel: false,
        border: {
            color: 'white'
        }
    },
    paletteSettings: {
        palette: [
            { value: 0, color: 'rgb(238,238,238)', label: 'no
contributions' },
            { value: 1, color: 'rgb(172, 213, 242)', label: '1-15
contributions' },
            { value: 16, color: 'rgb(127, 168, 201)', label: '16-31
contributions' },
            { value: 32, color: 'rgb(82, 123, 160)', label: '31-49
contributions' },
            { value: 50, color: 'rgb(37, 78, 119)', label: '50+
contributions' },
        ],
        type: 'Fixed',
        emptyPointColor: 'white'
    },
    showTooltip: true,
    legendSettings: {
        position: 'Bottom',
        width: '20%',
        alignment: 'Near',
        showLabel: true,
        labelDisplayType: 'None',
        enableSmartLegend: true
    },
    tooltipRender: (args: ITooltipEventArgs) => {
        let intl: Internationalization = new Internationalization();

```

```

    let format: Function = intl.getDateFormat({ format: 'EEE MMM dd,
YYYY' });
    let newDate : Date = <Date>args.xValue;
    let date: Date = new Date(newDate.getTime());
    let axisLabel: string[] =
args.heatmap.axisCollections[1].axisLabels;
    let index: number = axisLabel.indexOf(args.yLabel);
    (date).setDate((date).getDate() + index);
    let value: string = format(date);
    args.content = [(args.value === 0 ? 'No' : args.value) + ' ' +
'contributions' + '<br>' + value];
  },
  dataSource: heatmapData
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Multilevel Labels

Multilevel labels are used to classify a group of axis labels as a single category, which is then displayed with a label. By using [multiLevelLabels](#), you can add multiple levels on top of the axis labels.

To divide and group the axis labels, you can use [multiLevelLabels](#) property. The starting and ending indexes of the axis labels can be set using the [start](#) and [end](#) properties in the [categories](#). The [text](#) property can be used to specify a name for the grouped axis labels.

The multilevel labels can be customized by using the following properties.

- [overflow](#) - It is used to trim or wrap the multilevel labels when the label overflows the intended space. NOTE: This property is only for x-axis.
- [alignment](#) - It is used to place and align the multilevel labels.
- [maximumTextWidth](#) - It is used to set the maximum width of the text. When the text length exceeds the maximum text width, the overflow action will be performed.
- [textStyle](#) - It is used to customize the font style of the multilevel labels.
- [border](#) - It is used to customize the border of the multilevel labels displayed in the x-axis and y-axis.

INDEX.TS

```
import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [52, 65, 67, 45, 37, 52, 32, 76, 60, 64, 82, 91],
    [68, 52, 63, 51, 30, 51, 51, 81, 70, 60, 88, 80],
    [60, 50, 42, 53, 66, 70, 41, 69, 76, 74, 86, 97],
    [66, 64, 46, 40, 47, 41, 45, 76, 83, 69, 92, 84],
    [65, 42, 58, 32, 36, 44, 49, 79, 83, 69, 83, 93],
    [54, 46, 61, 46, 40, 39, 41, 69, 61, 84, 84, 87],
    [48, 46, 61, 47, 49, 41, 41, 67, 78, 83, 98, 87],
    [69, 52, 41, 44, 41, 52, 46, 71, 63, 84, 83, 91],
    [50, 59, 44, 43, 27, 42, 26, 64, 76, 65, 81, 86],
    [47, 49, 66, 53, 50, 34, 31, 79, 78, 79, 89, 95],
    [61, 40, 62, 26, 34, 54, 56, 74, 83, 78, 95, 98]
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Product wise Monthly sales revenue for a e-commerce website',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Laptop', 'Mobile', 'Gaming', 'Cosmetics', 'Fragrance',
            'Watches', 'Handbags', 'Apparels',
            'Kitchenware', 'Furniture', 'Home Decor'],
        multiLevelLabels: [
            {
                overflow: 'Trim',
                alignment: 'Near',
                textStyle: {
                    color: 'black',
                    fontWeight: 'Bold'
                }
            },
            { type: 'Rectangle', color: '#a19d9d' },
            categories: [
                { start: 0, end: 2, text: 'Electronics', },
                { start: 3, end: 4, text: 'Beauty and personal
care'},
                { start: 5, end: 7, text: 'Fashion', },
                { start: 8, end: 10, text: 'Household', }
```

```

    ],
    },
    ],
    },
    yAxis: {
        labels: ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
'Sep', 'Oct', 'Nov', 'Dec'],
        multiLevelLabels: [
            {
                border: { type: 'Brace', color: '#a19d9d' },
                categories: [
                    { start: 0, end: 2, text: 'Q1' },
                    { start: 3, end: 5, text: 'Q2' },
                    { start: 6, end: 8, text: 'Q3' },
                    { start: 9, end: 11, text: 'Q4' }
                ]
            },
            {
                border: { type: 'Brace', color: '#a19d9d' },
                categories: [
                    { start: 0, end: 5, text: 'First Half Yearly' },
                    {
                        start: 6,
                        end: 11,
                        text: 'Second Half Yearly'
                    }
                ]
            },
            {
                border: { type: 'Brace', color: '#a19d9d' },
                categories: [
                    { start: 0, end: 11, text: 'Yearly' }
                ]
            }
        ],
        legendSettings: {
            visible: false
        },
        paletteSettings: {
            palette: [{ color: '#F0C27B' },
                { color: '#4B1248' }
            ]
        },
        dataSource: heatmapData
    }); heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Palette in EJ2 JavaScript Heatmap chart control

In heat map, each data point is displayed as a cell with applied color based on the data value. The palette in the heat map is used to define the color range for cells and gradient type for colors. You can define the colors either in RGB or hex codes using the [color](#) property in the `palette`. The defined colors are applied to the cell background based on the palette type and cell value.

Palette types

You can display the heat map cells either in gradient colors or fixed colors.

Gradient

The smooth transition between the given palette colors can be applied for the heat map cells based on value. The heat map calculates all the gradient colors between the start and end colors for all distinct data values. Default start color and end color will be considered for gradient calculation, if the colors are not defined. The palette type must be defined as **Gradient** for the [type](#) property in `paletteSettings` property.

INDEX.TS

```

import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({

```

```

    titleSettings: {
      text: 'Sales Revenue per Employee (in 1000 US$)',
      textStyle: {
        size: '15px',
        fontWeight: '500',
        fontStyle: 'Normal',
        fontFamily: 'Segoe UI'
      }
    },
    xAxis: {
      labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
        'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
        'Mario'],
    },
    yAxis: {
      labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    paletteSettings: {
      palette: [
        { color: '#C06C84' },
        { color: '#6C5B7B' },
        { color: '#355C7D' }
      ],
      type: "Gradient"
    },
    dataSource: heatmapData,
    legendSettings: {
      visible: true,
    }
  });
  heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```



```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Fixed

In fixed palette type, solid colors are applied to the heat map cells. The data values can be grouped based on the number of colors defined for the heat map. The palette type should be defined as **Fixed** for the [type](#) property in the `paletteSettings` property.

INDEX.TS

```

import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
      'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
      'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  paletteSettings: {
    palette: [
      { color: '#C06C84' },
      { color: '#6C5B7B' },
      { color: '#355C7D' }
    ],
    type: "Fixed"
  },
  dataSource: heatmapData,
  legendSettings: {

```

```

        visible: true,
    }
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Defining color stops

You can define the colors ranges or color stops for data values in both gradient and fixed palette types. You need to define the data value in the **value** property for **palette** property to calculate the color stops. The heat map automatically calculates the color stops if the **value** property is is not defined. The **label** property is used to provide the additional information about the color that is to be displayed in the legend. If the label is not provided, the **value** is displayed in the legend. The labels can be automatically calculated based on data values, if both the values and labels are not defined.

INDEX.TS

```

import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
```

```

[56, 69, 21, 86, 3, 33],
[45, 7, 53, 81, 95, 79],
[60, 77, 74, 68, 88, 51],
[25, 25, 10, 12, 78, 14],
[25, 56, 55, 58, 12, 82],
[74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  paletteSettings: {
    palette: [
      { color: '#C06C84', label: 'Low', value: 50 },
      { color: '#6C5B7B', label: 'Moderate', value: 80 },
      { color: '#355C7D', label: 'High', value: 100 }
    ],
    type: "Gradient"
  },
  dataSource: heatmapData,
  legendSettings: {
    visible: true,
  }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Color range

The color range support is used to provide a specific color for specific range in heat map. The `startValue` and `endValue` properties are used to define the range start and end value. The `minColor` and `maxColor` properties represent the colors of given range. It's possible to set the cell color for the value not in the given range using the `fillColor` property.

In Fixed type, the `minColor` value is used for cell color.

INDEX.TS

```

import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 1],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
      'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
      'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],

```

```

    },
    paletteSettings: {
      palette: [
        { startValue:1, endValue:30, minColor: '#C2E7EC', maxColor:
'#AEDFE6' },
        { startValue:30, endValue:60, minColor: '#9AD7E0', maxColor:
'#72C7D4' },
        { startValue:60, endValue:90, minColor: '#5EBFCE', maxColor:
'#4AB7C8' }
      ],
      type: "Gradient"
    },
    dataSource: heatmapData,
    legendSettings: {
      visible: true,
    }
  });
  heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to enable smart legend](#)

Legend in EJ2 JavaScript Heatmap chart control

The legend is used to provide the information about the heat map cell. You can enable the legend by setting the [visible](#) property to **true** and injecting the **Legend** module using the **HeatMap.Inject(Legend)**.

INDEX.TS

```
import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    cellSettings: {
        showLabel: false,
    },
    paletteSettings: {
        palette: [
            { value: 0, color: '#C2E7EC' },
            { value: 10, color: '#AEDFE6' },
            { value: 20, color: '#9AD7E0' },
            { value: 30, color: '#72C7D4' },
            { value: 40, color: '#5EBFCE' },
            { value: 50, color: '#4AB7C8' },
            { value: 60, color: '#309DAE' },
            { value: 70, color: '#2B8C9B' },
            { value: 80, color: '#206974' },
            { value: 90, color: '#15464D' },
        ]
    }
});
```

```

        { value: 100, color: '#000000' },
    ],
  },
  dataSource: heatmapData,
  legendSettings: {
    position: 'Right',
  }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend types

Heat map supports two legend types: Gradient and list type.

- Gradient: This is a continuous color legend with smooth color transition between palette color values.
- List: List is a fixed color legend. Each palette color information is shown separately in the list item.

You can change the legend type by using the [type](#) property in the `paletteSettings` property.

INDEX.TS

```

import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);

```

```

let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    cellSettings: {
        showLabel: false,
    },
    paletteSettings: {
        type: 'Fixed'
    },
    dataSource: heatmapData,
    legendSettings: {
        position: 'Right',
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Placement

You can place the legend at left, right, top, or bottom to the heat map layout by using the [position](#) property. The legend is positioned at the right to the heat map by default.

INDEX.TS

```

import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {

```

```

        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
    legendSettings: {
        position: 'Top',
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Alignment

You can align the legend as center, far, or near to the heat map using the [alignment](#) property.

INDEX.TS

```

import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],

```

```

[25, 25, 10, 12, 78, 14],
[25, 56, 55, 58, 12, 82],
[74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
      'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  dataSource: heatmapData,
  legendSettings: {
    position: 'Right',
    alignment: 'Near',
    height: '150px'
  }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Legend dimensions

You can change the legend dimensions with values in pixels or percentage by using the [width](#) and [height](#) properties.

INDEX.TS

```
import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
    legendSettings: {
        position: 'Right',
        height: '150px',
    }
});
heatmap.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Paging for legend

Paging is available only for the list type legend in the heat map, and it can be enabled by default, when the legend items exceed the legend bounds. You can view each legend items by navigating between the pages using navigation buttons.

INDEX.TS

```

import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',

```

```

        fontFamily: 'Segoe UI'
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    paletteSettings: {
        palette: [
            { value: 0, color: '#C2E7EC' },
            { value: 10, color: '#AEDFE6' },
            { value: 20, color: '#9AD7E0' },
            { value: 25, color: '#86CFDA' },
            { value: 30, color: '#72C7D4' },
            { value: 40, color: '#5EBFCE' },
            { value: 50, color: '#4AB7C8' },
            { value: 55, color: '#36AFC2' },
            { value: 60, color: '#309DAE' },
            { value: 70, color: '#2B8C9B' },
            { value: 75, color: '#257A87' },
            { value: 80, color: '#206974' },
            { value: 85, color: '#1B5761' },
            { value: 90, color: '#15464D' },
            { value: 100, color: '#000000' },
        ],
        type: 'Fixed'
    },
    dataSource: heatmapData,
    legendSettings: {
        position: 'Right',
        height: "150px"
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use legend feature, we need to inject **Legend** using **HeatMap.Inject(Legend)**.

Smart Legend

Smart legend is another way of showing list type legend with responsiveness and readability, when the palette has more number of items. You can enable this smart legend by using the [enableSmartLegend](#) property when the palette type is set to **Fixed**.

In smart legend, you can change the display type of legend labels by using the [labelDisplayType](#) property.

The following are the legend label display types:

- All: Displays all labels in the legend.
- Edge: Displays the legend labels only at extreme ends.
- None: None of the labels are displayed. The tooltip will appear for this type of label display when hovering over the legend item.

INDEX.TS

```

import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  }
});

```

```

    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin', 'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    cellSettings: {
        showLabel: false,
    },
    paletteSettings: {
        palette: [
            { value: 0, color: '#C2E7EC' },
            { value: 10, color: '#AEDFE6' },
            { value: 20, color: '#9AD7E0' },
            { value: 30, color: '#72C7D4' },
            { value: 40, color: '#5EBFCE' },
            { value: 50, color: '#4AB7C8' },
            { value: 60, color: '#309DAE' },
            { value: 70, color: '#2B8C9B' },
            { value: 80, color: '#206974' },
            { value: 90, color: '#15464D' },
            { value: 100, color: '#000000' },
        ],
        type: 'Fixed'
    },
    dataSource: heatmapData,
    legendSettings: {
        position: 'Bottom',
        width: '75%',
        enableSmartLegend: true
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">

```



```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Selection

In the HeatMap, the legend selection is used to toggle the visibility of cell for view the specific range value. You can enable the legend selection using [toggleVisibility](#).

INDEX.TS

```

import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin', 'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    cellSettings: {
        showLabel: false,
    },
    paletteSettings: {
        palette: [
            { value: 0, color: '#C2E7EC' },
            { value: 10, color: '#AEDFE6' },

```

```

        { value: 20, color: '#9AD7E0' },
        { value: 30, color: '#72C7D4' },
        { value: 40, color: '#5EBFCE' },
        { value: 50, color: '#4AB7C8' },
        { value: 60, color: '#309DAE' },
        { value: 70, color: '#2B8C9B' },
        { value: 80, color: '#206974' },
        { value: 90, color: '#15464D' },
        { value: 100, color: '#000000' },
    ],
    type: 'Fixed'
},
dataSource: heatmapData,
legendSettings: {
    position: 'Bottom',
    width: '75%',
    enableSmartLegend: true,
    toggleVisibility: true
}
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Legend Title

The legend title displays a specific information about the legend. You can enable the legend title by setting the [title](#) property by providing the text and customizing the legend title text style using the [textStyle](#) property.

INDEX.TS

```
import { HeatMap, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee'
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    cellSettings: {
        showLabel: false,
    },
    paletteSettings: {
        palette: [
            { value: 0, color: '#6EB5D0' },
            { value: 50, color: '#7EDCA2' },
            { value: 100, color: '#DCD57E' },
        ],
    },
    dataSource: heatmapData,
    legendSettings: {
        position: 'Right',
        title: {
            text: "1000 US$"
        }
    }
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Appearance in EJ2 JavaScript HeatMap chart control

Cell customization

You can customize the cell by using the [cellSettings](#) property.

Border

Change the width, color, and radius of the heat map cells by using the [border](#) property.

INDEX.TS

```

import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {

```

```

        size: '15px',
        fontWeight: '500',
        fontStyle: 'Normal',
        fontFamily: 'Segoe UI'
    },
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
        'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
    cellSettings: {
        border: {
            width: 1,
            radius: 4,
            color: 'white'
        }
    },
    },
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Cell highlighting

Enable or disable the cell highlighting while hovering over the heat map cells by using the [enableCellHighlighting](#) property.

Note: The cell highlighting only works in a SVG rendering mode.

INDEX.TS

```
import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
    cellSettings: {
        enableCellHighlighting: true
    },
});
heatmap.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Color gradient mode

The [colorGradientMode](#) property can be used to set the minimum and maximum values for colors based on row and column. Three types of color gradient modes are available.

- **Table:** The minimum and maximum value colors calculated for overall data.
- **Row:** The minimum and maximum value colors calculated for each row of data.
- **Column:** The minimum and maximum value colors calculated for each column of data.

Note: The default value of `colorGradientMode` is **Table**.

INDEX.TS

```

import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',

```

```

        fontStyle: 'Normal',
        fontFamily: 'Segoe UI'
    },
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
    cellSettings: {
        enableCellHighlighting: true
    },
    paletteSettings: {
        colorGradientMode: 'Column'
    }
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Background color

The background color of the HeatMap can be customized using the [backgroundColor](#) property.

INDEX.TS

```
import { HeatMap, Tooltip, ICellEventArgs } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 1],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    backgroundColor: '#c7afcf',
    titleSettings: {
        text: 'Sales Revenue per Employee (in US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData
}); heatmap.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
      <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Margin

Set the margin for the HeatMap from its container by using the [margin](#) property.

INDEX.TS

```

import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  margin: { left: 15, right: 15, top: 15, bottom: 15 },
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
      'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
      'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  dataSource: heatmapData,
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Title

The title is used to provide a quick information about the data plotted in HeatMap. The [text](#) property is used to set the title for the HeatMap. The text style of the title can be customized by using the [textStyle](#) property.

INDEX.TS

```

import { HeatMap, Legend, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {

```

```

        size: '15px',
        fontWeight: '500',
        fontStyle: 'Italic',
        fontFamily: 'Segoe UI'
    },
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Data label

The visibility of data labels can be toggled using the [showLabel](#) property. By default, the data labels will be visible.

INDEX.TS

```
import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
```

```

HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
    cellSettings: {
        showLabel: false
    },
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="element"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the data label

The label displayed in the HeatMap cell can be changed using the [cellRender](#) event.

INDEX.TS

```

import { HeatMap, Tooltip, ICellEventArgs } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 1],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
      'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
      'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  dataSource: heatmapData,
  cellRender: (args: ICellEventArgs) => {
    args.displayText = (args.value as number) + 'K';
  },
}); heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Text style

The text attributes of the data label such as font-family, font-size, and color can be customized using the [textStyle](#) in the [cellSettings](#) property.

INDEX.TS

```

import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {

```

```

        size: '15px',
        fontWeight: '500',
        fontStyle: 'Normal',
        fontFamily: 'Segoe UI'
    },
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
    cellSettings: {
        textStyle: {
            fontStyle: 'Italic',
            fontFamily: 'Segoe UI'
        }
    },
    },
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```


Format

The format of the data label, such as currency, decimal, percent etc. can be changed using [format](#) property.

INDEX.TS

```
import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
    cellSettings: {
        format: '{value} $'
    },
});
heatmap.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Template

Any HTML elements can be added as a template in the data labels by using the [labelTemplate](#) property of [cellSettings](#) in the HeatMap.

The following examples show various data binding methods in the HeatMap using the [labelTemplate](#) property.

Array binding

By including `${value}` in the template content, the value from the data source for the corresponding cell can be displayed in the HeatMap cell as data label template content. Additionally, the x-axis and y-axis label values can be displayed by including `${xLabel}` and `${yLabel}` in the template content.

Table

The following example demonstrates how to add a data label template for array table binding.

INDEX.TS

```

import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any[] = [
    [[4, 39], [3, 8], [1, 3], [1, 10], [4, 4], [2, 15]],
    [[4, 28], [5, 92], [5, 73], [3, 1], [3, 4], [4, 126]],
    [[4, 45], [5, 152], [0, 44], [4, 54], [5, 243], [2, 45]]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Commercial Aviation Accidents and Fatalities by year 2015 -
2017',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'inherit',
        },
    },
    xAxis: {

```

```

        labels: ['2015', '2016', '2017']
    },
    yAxis: {
        labels: ['Jan-Feb', 'Mar-Apr', 'May-Jun', 'Jul-Aug', 'Sep-Oct',
'Nov-Dec']
    },
    cellSettings: {
        labelTemplate:
            '<div style="width:25px;height:20px;text-align:center;padding-
top:2px;background-color:#5BBB9C; border: 1px solid #000000; border-
radius:50%;font-weight:bold;">${value}</div>',
    },
    dataSource: heatmapData
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell

The following example demonstrates how to add a data label template for array cell binding.

INDEX.TS

```

import { HeatMap, Adaptor, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Adaptor, Legend);
let heatmapData: any[] = [

```

```

    [0, 0, [4, 39]], [0, 1, [3, 8]], [0, 2, [1, 3]], [0, 3, [1, 10]], [0, 4,
    [4, 4]], [0, 5, [2, 15]],
    [1, 0, [4, 28]], [1, 1, [5, 92]], [1, 2, [5, 73]], [1, 3, [3, 1]], [1,
    4, [3, 4]], [1, 5, [4, 126]],
    [2, 0, [4, 45]], [2, 1, [5, 152]], [2, 2, [0, 44]], [2, 3, [4, 54]], [2,
    4, [5, 243]], [2, 5, [2, 45]]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Commercial Aviation Accidents and Fatalities by year 2015 -
2017',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'inherit',
        },
    },
    xAxis: {
        labels: ['2015', '2016', '2017']
    },
    yAxis: {
        labels: ['Jan-Feb', 'Mar-Apr', 'May-Jun', 'Jul-Aug', 'Sep-Oct',
'Nov-Dec']
    },
    cellSettings: {
        labelTemplate:
            '<div style="width:25px;height:20px;text-align:center;padding-
top:2px;background-color:#5BBB9C; border: 1px solid #000000; border-
radius:50%;font-weight:bold;">${value}</div>',
    },
    dataSourceSettings: {
        isJsonData: false,
        adaptorType: 'Cell'
    },
    dataSource: heatmapData
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

JSON binding

By including the desired field name in the template content, such as `${value}`, the value from the data source for the corresponding cell can be displayed in the HeatMap cell as data label template content.

Table

The following example demonstrates how to add a data label template for JSON table binding.

INDEX.TS

```

import { HeatMap, Tooltip, Adaptor, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Adaptor, Legend);
let heatmapData: any[] = [
  {
    Year: '2017',
    image:
      'https://ej2.syncfusion.com/demos/src/circular-gauge/images/golfball.png',
    'Jan-Feb': [4, 39],
    'Mar-Apr': [3, 8],
    'May-Jun': [1, 3],
    'Jul-Aug': [1, 10],
    'Sep-Oct': [4, 4],
    'Nov-Dec': [2, 15]
  },
  {
    Year: '2016',
    image:
      'https://ej2.syncfusion.com/demos/src/circular-gauge/images/basketball.png',
    'Jan-Feb': [4, 28],
    'Mar-Apr': [5, 92],
    'May-Jun': [5, 73],
    'Jul-Aug': [3, 1],
    'Sep-Oct': [3, 4],
    'Nov-Dec': [4, 126]
  },
  {
    Year: '2015',
    image:
      'https://ej2.syncfusion.com/demos/src/circular-gauge/images/football.png',
    'Jan-Feb': [4, 45],
    'Mar-Apr': [5, 152],
    'May-Jun': [0, 44],

```

```

        'Jul-Aug': [4, 54],
        'Sep-Oct': [5, 243],
        'Nov-Dec': [2, 45]
    },
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Commercial Aviation Accidents and Fatalities by year 2015 -
2017',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'inherit'
        },
    },
    xAxis: {
        labels: ['2015', '2016', '2017']
    },
    yAxis: {
        labels: ['Jan-Feb', 'Mar-Apr', 'May-Jun', 'Jul-Aug', 'Sep-Oct',
'Nov-Dec'],
    },
    cellSettings: {
        labelTemplate:
            "<div><img style='width:20px;height:20px;' src='${image}' />
</div>"
    },
    paletteSettings: {
        palette: [{ color: '#C06C84' }, { color: '#6C5B7B' }, { color:
'#355C7D' }],
        type: 'Gradient',
    },
    dataSourceSettings: {
        isJsonData: true,
        adaptorType: 'Table',
        xDataMapping: 'Year'
    },
    dataSource: heatmapData
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell

The following example demonstrates how to add a data label template for JSON cell binding.

INDEX.TS

```

import { HeatMap, Tooltip, Adaptor, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Adaptor, Legend);
let heatmapData: any[] = [
    { Year: '2017', Months: 'Jan-Feb', Accidents: 4, Fatalities: 39 },
    { Year: '2017', Months: 'Mar-Apr', Accidents: 3, Fatalities: 8 },
    { Year: '2017', Months: 'May-Jun', Accidents: 1, Fatalities: 3 },
    { Year: '2017', Months: 'Jul-Aug', Accidents: 1, Fatalities: 10 },
    { Year: '2017', Months: 'Sep-Oct', Accidents: 4, Fatalities: 4 },
    { Year: '2017', Months: 'Nov-Dec', Accidents: 2, Fatalities: 15 },
    { Year: '2016', Months: 'Jan-Feb', Accidents: 4, Fatalities: 28 },
    { Year: '2016', Months: 'Mar-Apr', Accidents: 5, Fatalities: 92 },
    { Year: '2016', Months: 'May-Jun', Accidents: 5, Fatalities: 73 },
    { Year: '2016', Months: 'Jul-Aug', Accidents: 3, Fatalities: 1 },
    { Year: '2016', Months: 'Sep-Oct', Accidents: 3, Fatalities: 4 },
    { Year: '2016', Months: 'Nov-Dec', Accidents: 4, Fatalities: 126 },
    { Year: '2015', Months: 'Jan-Feb', Accidents: 4, Fatalities: 45 },
    { Year: '2015', Months: 'Mar-Apr', Accidents: 5, Fatalities: 152 },
    { Year: '2015', Months: 'May-Jun', Accidents: 0, Fatalities: 0 },
    { Year: '2015', Months: 'Jul-Aug', Accidents: 4, Fatalities: 54 },
    { Year: '2015', Months: 'Sep-Oct', Accidents: 5, Fatalities: 243 },
    { Year: '2015', Months: 'Nov-Dec', Accidents: 2, Fatalities: 45 },
];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Commercial Aviation Accidents and Fatalities by year 2015 -
2017',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'inherit',
        },
    },
    xAxis: {

```

```

        labels: ['2015', '2016', '2017'],
    },
    yAxis: {
        labels: ['Jan-Feb', 'Mar-Apr', 'May-Jun', 'Jul-Aug', 'Sep-Oct',
'Nov-Dec'],
    },
    cellSettings: {
        labelTemplate: '<div> Accidents - ${Accidents}</div>',
    },
    dataSourceSettings: {
        isJsonData: true,
        adaptorType: 'Cell',
        xDataMapping: 'Year',
        yDataMapping: 'Months',
        valueMapping: 'Fatalities',
    },
    dataSource: heatmapData,
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [To customize the appearance of tool tip](#)

Dimensions in EJ2 JavaScript Heatmap chart control

Size for container

Heat map can be rendered to its container size. You can set the size through inline or CSS.

```
`javascript
<div id='container'>

<div id='element' style="width:650px; height:350px;"></div>

</div>
`
```

Size for heat map

You can set the size of heat map directly by using the [width](#) and [height](#) properties.

In Pixel

You can set the size for heat map in a pixel.

INDEX.TS

```
import { HeatMap } from '@syncfusion/ej2-heatmap';
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    width: '650px',
    height: '350px',
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
});
```

```
heatmap.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

In percentage

By setting value in percentage, heat map gets its dimension with respect to its container. For example, when the height is '50%', heat map rendered to half of the container height.

INDEX.TS

```
import { HeatMap } from '@syncfusion/ej2-heatmap';
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  width: '80%',
  height: '90%',
```

```

    titleSettings: {
      text: 'Sales Revenue per Employee (in 1000 US$)',
      textStyle: {
        size: '15px',
        fontWeight: '500',
        fontStyle: 'Normal',
        fontFamily: 'Segoe UI'
      }
    },
    xAxis: {
      labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
        'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
        'Mario'],
    },
    yAxis: {
      labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
  });
  heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip in EJ2 JavaScript Heatmap chart control

Tooltip is used to provide the details of the heat map cell, and this can be displayed, while hovering the cursor over the cell or performing tap action in touch devices.

Default Tooltip

You can enable the tooltip by setting the [showTooltip](#) property to **true** and injecting the **Tooltip** module using the **HeatMap.Inject(Tooltip)**.

INDEX.TS

```
import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [0.72, 0.71, 0.71, 0.67, 0.72, 0.53, 0.53, 0.56, 0.58, 0.56],
    [2.28, 2.29, 2.09, 1.84, 1.64, 1.49, 1.49, 1.39, 1.32, 1.23],
    [2.02, 2.17, 2.30, 2.39, 2.36, 2.52, 2.62, 2.57, 2.57, 2.74],
    [3.21, 3.26, 3.45, 3.47, 3.42, 3.34, 3.14, 2.83, 2.64, 2.61],
    [3.22, 3.13, 3.04, 2.95, 2.69, 2.49, 2.27, 2.18, 2.06, 1.87],
    [3.30, 3.39, 3.40, 3.48, 3.60, 3.67, 3.73, 3.79, 3.79, 4.07],
    [5.80, 5.74, 5.64, 5.44, 5.18, 5.08, 5.07, 5.00, 5.35, 5.47],
    [6.91, 7.40, 8.13, 8.80, 9.04, 9.24, 9.43, 9.35, 9.49, 9.69]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Crude Oil Production of Non-OPEC Countries (in Million
barrels per day)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Canada', 'China', 'Egypt', 'Mexico', 'Norway',
'Russia', 'UK', 'USA']
    },
    yAxis: {
        labels: ['2000', '2001', '2002', '2003', '2004', '2005',
'2006', '2007', '2008', '2009', '2010'],
    },
    cellSettings: {
        showLabel: false,
    },
    dataSource: heatmapData,
    showTooltip: true
});
heatmap.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip template

In heat map, you can customize the tooltip using the [tooltipRender](#) client-side event.

INDEX.TS

```

import { HeatMap, Tooltip, ITooltipEventArgs } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [0.72, 0.71, 0.71, 0.67, 0.72, 0.53, 0.53, 0.56, 0.58, 0.56],
    [2.28, 2.29, 2.09, 1.84, 1.64, 1.49, 1.49, 1.39, 1.32, 1.23],
    [2.02, 2.17, 2.30, 2.39, 2.36, 2.52, 2.62, 2.57, 2.57, 2.74],
    [3.21, 3.26, 3.45, 3.47, 3.42, 3.34, 3.14, 2.83, 2.64, 2.61],
    [3.22, 3.13, 3.04, 2.95, 2.69, 2.49, 2.27, 2.18, 2.06, 1.87],
    [3.30, 3.39, 3.40, 3.48, 3.60, 3.67, 3.73, 3.79, 3.79, 4.07],
    [5.80, 5.74, 5.64, 5.44, 5.18, 5.08, 5.07, 5.00, 5.35, 5.47],
    [6.91, 7.40, 8.13, 8.80, 9.04, 9.24, 9.43, 9.35, 9.49, 9.69]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Crude Oil Production of Non-OPEC Countries (in Million
barrels per day)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Canada', 'China', 'Egypt', 'Mexico', 'Norway',
'Russia', 'UK', 'USA']
    },
    yAxis: {
        labels: ['2000', '2001', '2002', '2003', '2004', '2005',
'2006', '2007', '2008', '2009', '2010'],
    },
    cellSettings: {

```

```

        showLabel: false,
    },
    dataSource: heatmapData,
    showTooltip: true,
    tooltipRender: (args: ITooltipEventArgs) => {
        args.content = ['In ' + args.yLabel + ', the ' + args.xLabel + '
produced ' + args.value + ' million barrels per day'];
    },
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the appearance of Tooltip

The [fill](#) and [border](#) properties are used to customize the background color and border of the tooltip respectively. The [textStyle](#) property in the tooltip is used to customize the font of the tooltip text.

INDEX.TS

```

import { HeatMap, Tooltip, ITooltipEventArgs } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
  [0.72, 0.71, 0.71, 0.67, 0.72, 0.53, 0.53, 0.56, 0.58, 0.56],
  [2.28, 2.29, 2.09, 1.84, 1.64, 1.49, 1.49, 1.39, 1.32, 1.23],
  [2.02, 2.17, 2.30, 2.39, 2.36, 2.52, 2.62, 2.57, 2.57, 2.74],
  [3.21, 3.26, 3.45, 3.47, 3.42, 3.34, 3.14, 2.83, 2.64, 2.61],

```

```

[3.22, 3.13, 3.04, 2.95, 2.69, 2.49, 2.27, 2.18, 2.06, 1.87],
[3.30, 3.39, 3.40, 3.48, 3.60, 3.67, 3.73, 3.79, 3.79, 4.07],
[5.80, 5.74, 5.64, 5.44, 5.18, 5.08, 5.07, 5.00, 5.35, 5.47],
[6.91, 7.40, 8.13, 8.80, 9.04, 9.24, 9.43, 9.35, 9.49, 9.69]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Crude Oil Production of Non-OPEC Countries (in Million
barrels per day)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Canada', 'China', 'Egypt', 'Mexico', 'Norway',
'Russia', 'UK', 'USA']
  },
  yAxis: {
    labels: ['2000', '2001', '2002', '2003', '2004', '2005',
'2006', '2007', '2008', '2009', '2010'],
  },
  paletteSettings: {
    palette: [{ color: '#F0ADCE' },
{ color: '#19307B' }
]
  },
  cellSettings: {
    showLabel: false,
  },
  dataSource: heatmapData,
  showTooltip: true,
  tooltipSettings: {
    fill: '#696295',
    textStyle: {
      color: 'FFFFFF',
      size: '12px'
    },
    border: {
      width: 2,
      color: '#F0C27B'
    }
  },
  tooltipRender: (args: ITooltipEventArgs) => {
    args.content = ['In ' + args.yLabel + ', the ' + args.xLabel + '
produced ' + args.value + ' million barrels per day'];
  },
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use tooltip feature, we need to inject **Tooltip** using **HeatMap.Inject(Tooltip)**.

Selection in EJ2 JavaScript Heatmap chart control

In the HeatMap, the cell selection is used to select single or multiple HeatMap cells at runtime and get the selected cell details using the [cellSelected](#) event. You can enable the cell selection using the [allowSelection](#) property.

The HeatMap cells can be selected using the following interactions, as shown in the table below.

Modes of Interactions	Description
Mouse	HeatMap cells can be selected by clicking or dragging and dropping over them.
Touch	HeatMap cells can be selected by tapping or dragging and dropping over them.
Keyboard	The Ctrl key on the keyboard can be used to enable multiple cell selection with mouse and touch interaction. The Ctrl key can only be used if the enableMultiSelect property is set to true in order to enable multiple cell selection.

INDEX.TS

```

import { HeatMap, Tooltip, ICellEventArgs } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip);
let heatmapData: any [] = [

```



```

        [73, 39, 26, 39, 94, 0],
        [93, 58, 53, 38, 26, 68],
        [99, 28, 22, 4, 66, 90],
        [14, 26, 97, 69, 69, 3],
        [7, 46, 47, 47, 88, 6],
        [41, 55, 73, 23, 3, 79],
        [56, 69, 21, 86, 3, 33],
        [45, 7, 53, 81, 95, 79],
        [60, 77, 74, 68, 88, 51],
        [25, 25, 10, 12, 78, 14],
        [25, 56, 55, 58, 12, 82],
        [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  allowSelection: true,
  dataSource: heatmapData,
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Enable single cell selection

In the HeatMap, the [enableMultiSelect](#) property is used to allow single cell selection. When you set the [enableMultiSelect](#) property to **false**, only one cell is selected. By default, [enableMultiSelect](#) property is set to **true**.

INDEX.TS

```
import { HeatMap } from '@syncfusion/ej2-heatmap';
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
      'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
      'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  allowSelection: true,
  enableMultiSelect: false,
  dataSource: heatmapData,
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Clearing cell selection

The [clearSelection](#) method can be used to clear all the selected cells. The below example illustrates the same.

INDEX.TS

```

import { HeatMap, Legend, Tooltip } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Legend, Tooltip);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',

```

```

        fontFamily: 'Segoe UI',
    },
},
xAxis: {
    labels: [
        'Nancy',
        'Andrew',
        'Janet',
        'Margaret',
        'Steven',
        'Michael',
        'Robert',
        'Laura',
        'Anne',
        'Paul',
        'Karin',
        'Mario',
    ],
},
yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
},
allowSelection: true,
dataSource: heatmapData,
});
heatmap.appendTo('#element');
const selectionElement = document.getElementById('selection');
if(selectionElement) {
    selectionElement.onclick = () => {
        heatmap.clearSelection();
    };
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <button id="selection">Clear Selection</button>
  </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in EJ2 JavaScript HeatMap chart control

HeatMap has built-in accessibility features like screen reading. Screen reading in the HeatMap control allows all users, regardless of ability or disability, to use the control. The following HeatMap elements will be read aloud with screen reading software like Narrator for Windows.

Elements	Description
---	---
Title	Reads the contents of the HeatMap chart's title.
Axis labels	Reads the x and y axis labels of the HeatMap chart.
Multilevel labels	Reads the multilevel labels in the x and y axis of the HeatMap chart.
Cell labels	Reads the labels from the cells in the Heatmap chart.
Legend title	Reads the contents of the legend's title as specified in HeatMap chart.
Legend item label	Reads the label of a legend item in HeatMap chart.

Ensuring accessibility

The HeatMap control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the HeatMap control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the HeatMap control with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript controls](#)

Events in EJ2 JavaScript HeatMap chart control

This section describes the HeatMap chart control event, which occurs when the required actions are performed.

cellClick

When you click on a HeatMap cell, the [cellClick](#) event is triggered. To know more about arguments of this event, refer [here](#).

INDEX.TS

```

import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any[] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],

```

```

[14, 26, 97, 69, 69, 3],
[7, 46, 47, 47, 88, 6],
[41, 55, 73, 23, 3, 79],
[56, 69, 21, 86, 3, 33],
[45, 7, 53, 81, 95, 79],
[60, 77, 74, 68, 88, 51],
[25, 25, 10, 12, 78, 14],
[25, 56, 55, 58, 12, 82],
[74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  cellClick: function(args) {
    console.log('The cell click event has been triggered!!!', args);
  },
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
      'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
      'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  dataSource: heatmapData,
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

cellDoubleClick

When you double click on a HeatMap cell, the [cellDoubleClick](#) event is triggered. To know more about arguments of this event, refer [here](#).

INDEX.TS

```

import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    cellDoubleClick: function(args) {
        console.log('The cell double click event has been triggered!!!',
args);
    },
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

cellRender

The [cellRender](#) event will be triggered before each HeatMap cell is rendered. To know more about arguments of this event, refer [here](#).

INDEX.TS

```

import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  cellRender: function(args) {
    console.log('The cell render event has been triggered!!!', args);
  },
  titleSettings: {

```



```

        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        },
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

cellSelected

When single or multiple cells in the HeatMap are selected, the [cellSelected](#) event is triggered. To know more about arguments of this event, refer [here](#).

INDEX.TS

```
import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    cellSelected: function(args) {
        console.log('The cell selected event has been triggered!!!', args);
    },
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    allowSelection: true,
    dataSource: heatmapData,
});
heatmap.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

created

Once HeatMap has been completely rendered, the [created](#) event is triggered.

INDEX.TS

```

import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    created: function(args) {
        console.log('The created event has been triggered!!!');
    },
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },

```

```

        dataSource: heatmapData,
    });
    heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

legendRender

The [legendRender](#) event is triggered before the legend is rendered. To know more about arguments of this event, refer [here](#).

INDEX.TS

```

import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
  [56, 69, 21, 86, 3, 33],
  [45, 7, 53, 81, 95, 79],
  [60, 77, 74, 68, 88, 51],
  [25, 25, 10, 12, 78, 14],
  [25, 56, 55, 58, 12, 82],
  [74, 33, 88, 23, 86, 59]];

```

```

let heatmap: HeatMap = new HeatMap({
  legendRender: function(args) {
    console.log('The legend render event has been triggered!!!', args);
  },
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
      'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
      'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  dataSource: heatmapData,
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

load

The [load](#) event is triggered before the HeatMap is rendered. To know more about arguments of this event, refer [here](#).

INDEX.TS

```
import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    load: function(args) {
        console.log('The load event has been triggered!!!', args);
    },
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
});
heatmap.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

loaded

Once HeatMap is loaded, the [loaded](#) event is triggered. To know more about arguments of this event, refer [here](#).

INDEX.TS

```

import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    loaded: function(args) {
        console.log('The loaded event has been triggered!!!', args);
    },
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',

```

```

        'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
        'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
});
heatmap.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

resized

When the window is resized, the [resized](#) event is triggered to notify the resize of the HeatMap. To know more about arguments of this event, refer [here](#).

INDEX.TS

```

import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any [] = [
  [73, 39, 26, 39, 94, 0],
  [93, 58, 53, 38, 26, 68],
  [99, 28, 22, 4, 66, 90],
  [14, 26, 97, 69, 69, 3],
  [7, 46, 47, 47, 88, 6],
  [41, 55, 73, 23, 3, 79],
```



```

[56, 69, 21, 86, 3, 33],
[45, 7, 53, 81, 95, 79],
[60, 77, 74, 68, 88, 51],
[25, 25, 10, 12, 78, 14],
[25, 56, 55, 58, 12, 82],
[74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
  resized: function(args) {
    console.log('The resized event has been triggered!!!', args);
  },
  titleSettings: {
    text: 'Sales Revenue per Employee (in 1000 US$)',
    textStyle: {
      size: '15px',
      fontWeight: '500',
      fontStyle: 'Normal',
      fontFamily: 'Segoe UI'
    }
  },
  xAxis: {
    labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
'Mario'],
  },
  yAxis: {
    labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
  },
  dataSource: heatmapData,
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

tooltipRender

The [tooltipRender](#) event is triggered before the tooltip is rendered on the HeatMap cell. To know more about arguments of this event, refer [here](#).

INDEX.TS

```

import { HeatMap, Tooltip, Legend } from '@syncfusion/ej2-heatmap';
HeatMap.Inject(Tooltip, Legend);
let heatmapData: any [] = [
    [73, 39, 26, 39, 94, 0],
    [93, 58, 53, 38, 26, 68],
    [99, 28, 22, 4, 66, 90],
    [14, 26, 97, 69, 69, 3],
    [7, 46, 47, 47, 88, 6],
    [41, 55, 73, 23, 3, 79],
    [56, 69, 21, 86, 3, 33],
    [45, 7, 53, 81, 95, 79],
    [60, 77, 74, 68, 88, 51],
    [25, 25, 10, 12, 78, 14],
    [25, 56, 55, 58, 12, 82],
    [74, 33, 88, 23, 86, 59]];
let heatmap: HeatMap = new HeatMap({
    tooltipRender: function(args) {
        console.log('The tooltip render event has been triggered!!!', args);
    },
    titleSettings: {
        text: 'Sales Revenue per Employee (in 1000 US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
            'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin',
            'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    dataSource: heatmapData,
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 HeatMap</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How To

Tooltip template in EJ2 JavaScript Heatmap chart control

You can show a tooltip as a table using the `template` property in `tooltipSettings`.

The following steps describe how to show the table tooltip.

Step 1:

Initialize the tooltip template div as shown in the following html page.

,

```
<script id="tooltipTemplate" type="text/x-template">
```

```
<div id='templateWrap'>
```

<code>\${xValue}:</code>	<code>\${yValue}</code>	<code>\${value}</code>
--------------------------	-------------------------	------------------------

```
</div>
```

```
</script>
```

,

Step 2:

Set the element id to the `template` property in `tooltipSettings` to show the tooltip template.

INDEX.TS

```
import { HeatMap, Tooltip } from '@syncfusion/ej2-heatmap';
```

```

HeatMap.Inject(Tooltip);
let heatmapData: any [] = [
    [0.72, 0.71, 0.71, 0.67, 0.72, 0.53, 0.53, 0.56, 0.58, 0.56],
    [2.28, 2.29, 2.09, 1.84, 1.64, 1.49, 1.49, 1.39, 1.32, 1.23],
    [2.02, 2.17, 2.30, 2.39, 2.36, 2.52, 2.62, 2.57, 2.57, 2.74],
    [3.21, 3.26, 3.45, 3.47, 3.42, 3.34, 3.14, 2.83, 2.64, 2.61],
    [3.22, 3.13, 3.04, 2.95, 2.69, 2.49, 2.27, 2.18, 2.06, 1.87],
    [3.30, 3.39, 3.40, 3.48, 3.60, 3.67, 3.73, 3.79, 3.79, 4.07],
    [5.80, 5.74, 5.64, 5.44, 5.18, 5.08, 5.07, 5.00, 5.35, 5.47],
    [6.91, 7.40, 8.13, 8.80, 9.04, 9.24, 9.43, 9.35, 9.49, 9.69]];
let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Crude Oil Production of Non-OPEC Countries (in Million
barrels per day)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Canada', 'China', 'Egypt', 'Mexico', 'Norway',
'Russia', 'UK', 'USA']
    },
    yAxis: {
        labels: ['2000', '2001', '2002', '2003', '2004', '2005',
'2006', '2007', '2008', '2009', '2010'],
    },
    cellSettings: {
        showLabel: false,
    },
    tooltipSettings: {
        fill: '#265259',
        textStyle: {
            color: 'FFFFFF',
            size: '12px'
        },
        border: {
            width: 1,
            color: '#98BABF'
        },
        template: "#tooltipTemplate"
    },
    dataSource: heatmapData,
    showTooltip: true,
    },
});
heatmap.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 HeatMap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="tooltipTemplate" type="text/x-template">
    <div id='templateWrap'>
      <table style="width:100%; border: 1px solid black;">
        <tr><td bgcolor="#00FFFF">${xValue}</td><td
bgcolor="#00FFFF">${yValue}</td><td bgcolor="#00FFFF">${value}</td></tr>
      </table>
    </div>
  </script>

  <div id="container">
    <div id="element">
    </div>
  </div>
  <style>
    table,
    th,
    td {
      border: 1px solid rgb(105, 105, 105);
      border-collapse: collapse;
    }
  </style>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend customization in EJ2 JavaScript Heatmap chart control

You can change the legend label using the `legendRender` client-side event. You can also hide the legend label using this client-side event.

INDEX.TS

```

import { HeatMap, Legend, ILegendRenderEventArgs } from '@syncfusion/ej2-heatmap';
import { EmitType } from '@syncfusion/ej2-base';
HeatMap.Inject(Legend);
let heatmapData: any[] = [
  [73000, 39000, 26000, 39000, 94000, 0],
  [93000, 58000, 53000, 38000, 26000, 68000],
  [99000, 28000, 22000, 4000, 66000, 9000],
  [14000, 26000, 97000, 69000, 69000, 3000],

```

```

[7000, 46000, 47000, 47000, 88000, 6000],
[41000, 55000, 73000, 23000, 30000, 79000],
[56000, 69000, 21000, 86000, 3000, 33000],
[45000, 7000, 53000, 81000, 95000, 79000],
[60000, 77000, 74000, 68000, 88000, 51000],
[25000, 25000, 10000, 12000, 78000, 14000],
[25000, 56000, 55000, 58000, 12000, 82000],
[74000, 33000, 88000, 23000, 86000, 59000]];

let legendRender: EmitType<ILegendRenderEventArgs> = (args:
ILegendRenderEventArgs): void => {
    if(args.text=='25,000' || args.text=='50,000' ||
args.text=='99,000'){
        args.text = args.text.replace(/,//, "");
        args.text = ` ${parseInt(args.text/1000)} ` + "k "+"$";
    } else {
        args.cancel=true;
    }
};

let heatmap: HeatMap = new HeatMap({
    titleSettings: {
        text: 'Sales Revenue per Employee (in US$)',
        textStyle: {
            size: '15px',
            fontWeight: '500',
            fontStyle: 'Normal',
            fontFamily: 'Segoe UI'
        }
    },
    xAxis: {
        labels: ['Nancy', 'Andrew', 'Janet', 'Margaret', 'Steven',
        'Michael', 'Robert', 'Laura', 'Anne', 'Paul', 'Karin', 'Mario'],
    },
    yAxis: {
        labels: ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
    },
    cellSettings: {
        showLabel: false,
    },
    legendRender: legendRender,
    paletteSettings: {
        palette: [
            { value: 0, color: '#C2E7EC' },
            { value: 25000, color: '#AEDFE6' },
            { value: 50000, color: '#9AD7E0' },
            { value: 75000, color: '#72C7D4' },
            { value: 99000, color: '#5EBFCE' },
        ],
        type: 'Gradient'
    },
    showTooltip: true,
    dataSource: heatmapData,
});
heatmap.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 HeatMap</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
  <script id="tooltipTemplate" type="text/x-template">
    <div id='templateWrap'>
      <table style="width:100%; border: 1px solid black;">
        <tr><td bgcolor="#00FFFF">${xValue}</td><td
        bgcolor="#00FFFF">${yValue}</td><td bgcolor="#00FFFF">${value}</td></tr>
      </table>
    </div>
  </script>

  <div id="container">
    <div id="element">
    </div>
  </div>
  <style>
    table,
    th,
    td {
      border: 1px solid rgb(105, 105, 105);
      border-collapse: collapse;
    }
  </style>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Ej1 api migration in EJ2 JavaScript Heatmap chart control

This article describes the API migration process of heat map component from Essential JS 1 to Essential JS 2.

Members

<!-- markdownlint-disable MD033 -->

Beha viour	API in Essential JS 1	API in Essential JS 2
---------------	-----------------------	-----------------------

Specifies the width of the heatmap	Property:width \$("#heatmap").ejHeatMap({ width: "300"});	Property:width var heatmap = new ej.heatmap.HeatMap({ width: '650px'}); heatmap.appendTo('#heatmap');
Specifies the height of the heatmap	Property:height \$("#heatmap").ejHeatMap({ height: "300"});	Property:height var heatmap = new ej.heatmap.HeatMap({ height: '650px'}); heatmap.appendTo('#heatmap');
Enables or disables tooltip of heatmap	Property:showtooltip \$("#heatmap").ejHeatMap({ showtooltip: true});	Property:showTooltip var heatmap = new ej.heatmap.HeatMap({ showTooltip: true}); heatmap.appendTo('#heatmap');
Defines the tooltip that should be shown when the mouse hovers over cells.	Property:toolTipSettings.templateId \$("#heatmap").ejHeatMap({ toolTipSettings: { templateId: "mouseoverToolTipId" }});	Property:tooltipRender var heatmap = new ej.heatmap.HeatMap({ tooltipRender: function (args) { args.content = [args.yLabel + ' ' + args.xLabel + ' : ' + args.value]; }}); heatmap.appendTo('#heatmap');
Specifies the	Property:itemsSource \$("#heatmap").ejHeatMap({ itemsSource: []});	Property:dataSource var heatmap = new ej.heatmap.HeatMap({ dataSource:

source data of the heatmap.		<pre>[]});heatmap.appendTo('#heatmap');</pre>
Specifies whether the cell content can be visible or not.	<pre>Property:heatmapCell.showContent\$("#heatmap").ejHeatMap({ heatmapCell: { showContent: ej.HeatMap.CellVisibility.Hidden } });</pre>	<pre>Property:cellSettings.showLabelvar heatmap = new ej.heatmap.HeatMap({ cellSettings: { showLabel: false }, });heatmap.appendTo('#heatmap');</pre>
Specifies the color of the heatmap column data.	<pre>Property:colorMappingCollection.color\$("#heatmap").ejHeatMap({ colorMappingCollection: [{ color: "#8ec8f8" }, { color: "#0d47a1" }] });</pre>	<pre>Property:paletteSettings.palette.colorvar heatmap = new ej.heatmap.HeatMap({ paletteSettings: { palette: [{ color: '#C06C84'},] } });heatmap.appendTo('#heatmap');</pre>
Specifies the color values of the heatmap column data.	<pre>Property:colorMappingCollection.value\$("#heatmap").ejHeatMap({ colorMappingCollection: [{ value: 0 }, { value: 100 }] });</pre>	<pre>Property:paletteSettings.palette.valuevar heatmap = new ej.heatmap.HeatMap({ paletteSettings: { palette: [{ value: 50 }, { value: 100 }] } });heatmap.appendTo('#heatmap');</pre>

Specifies the label text of the heatmap color.	Property:colorMappingCollection.label.text\$("#heatmap").ejHeatMap({ colorMappingCollection: [{ label: { text: "Low" } }, { label: { text: "Moderate" } }] });	Property:paletteSettings.palette.label var heatmap = new ej.heatmap.HeatMap({ paletteSettings: { palette: [{ label:'Low' }, { label:'Moderate' }] } });heatmap.appendTo('#heatmap');
Specifies the style of the heatmap color label.	Property:colorMappingCollection.label.boldProperty:colorMappingCollection.label.italic\$("#heatmap").ejHeatMap({ colorMappingCollection: [{ label: { bold: true } }, { label: { italic: true } }], });	Property:legendSettings.textStyle.fontStyle var heatmap = new ej.heatmap.HeatMap({ legendSettings: { textStyle: { fontStyle:'bold' } } });heatmap.appendTo('#heatmap');
Specifies the font size of the heatmap label.	Property:colorMappingCollection.label.fontSize\$("#heatmap").ejHeatMap({ colorMappingCollection: [{ label: { fontSize: 18 } }] });	Property:legendSettings.textStyle.size var heatmap = new ej.heatmap.HeatMap({ legendSettings: { textStyle: { size: 18 } } });heatmap.appendTo('#heatmap');
Specifies the font family of the heatmap label.	Property:colorMappingCollection.label.fontFamily\$("#heatmap").ejHeatMap({ colorMappingCollection: [{ label: { fontFamily: "Arial" } }] });	Property:legendSettings.textStyle.fontFamily var heatmap = new ej.heatmap.HeatMap({ legendSettings: { textStyle: { fontFamily: 'Arial' } } });heatmap.appendTo('#heatmap');
Specifies the font color	Property:colorMappingCollection.label.fontColor\$("#heatmap").ejHeatMap({ colorMappingCollection: [{ label: { fontColor: "red" } }] });	Property:legendSettings.textStyle.fontFamily var heatmap = new ej.heatmap.HeatMap({ legendSettings: { textStyle: { color:

of the heatmap label.		'red' } });heatmap.appendTo ('#heatmap');
Specifies the mapping name of the column.	Property:itemsMapping.column.propertyName\$("#heatmap").ejHeatMap({ itemsMapping: { column: { "propertyName": "ProductName" } } });	Property:dataSource.yDataMappingvar heatmap = new ej.heatmap.HeatMap({ dataSource: heatmapData, dataSourceSettings: { yDataMapping: 'columnid' } });heatmap.appendTo ('#heatmap');
Specifies the mapping name of the row.	Property:itemsMapping.row.propertyName\$("#heatmap").ejHeatMap({ itemsMapping: { row: { "displayName": "Product Name" } } });	Property:dataSource.xDataMappingvar heatmap = new ej.heatmap.HeatMap({ dataSource: heatmapData, dataSourceSettings: { xDataMapping: 'rowid' } });heatmap.appendTo ('#heatmap');
Specifies the mapping name of the row.	Property:itemsMapping.value.displayName\$("#heatmap").ejHeatMap({ itemsMapping: { value: { "displayName": "Product Name" } } });	Property:dataSource.valueMappingvar heatmap = new ej.heatmap.HeatMap({ dataSource: heatmapData, dataSourceSettings: { valueMapping: 'value' } });heatmap.appendTo ('#heatmap');

Events

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Triggered when the cell get clicked.	Property:cellSelected\$("#heatmap").ejHeatMap({ cellSelected: function(args) {} });	Property:cellClickvar heatmap = new ej.heatmap.HeatMap({ cellClick: function (args) {}, });heatmap.appendTo ('#heatmap');

Image Editor

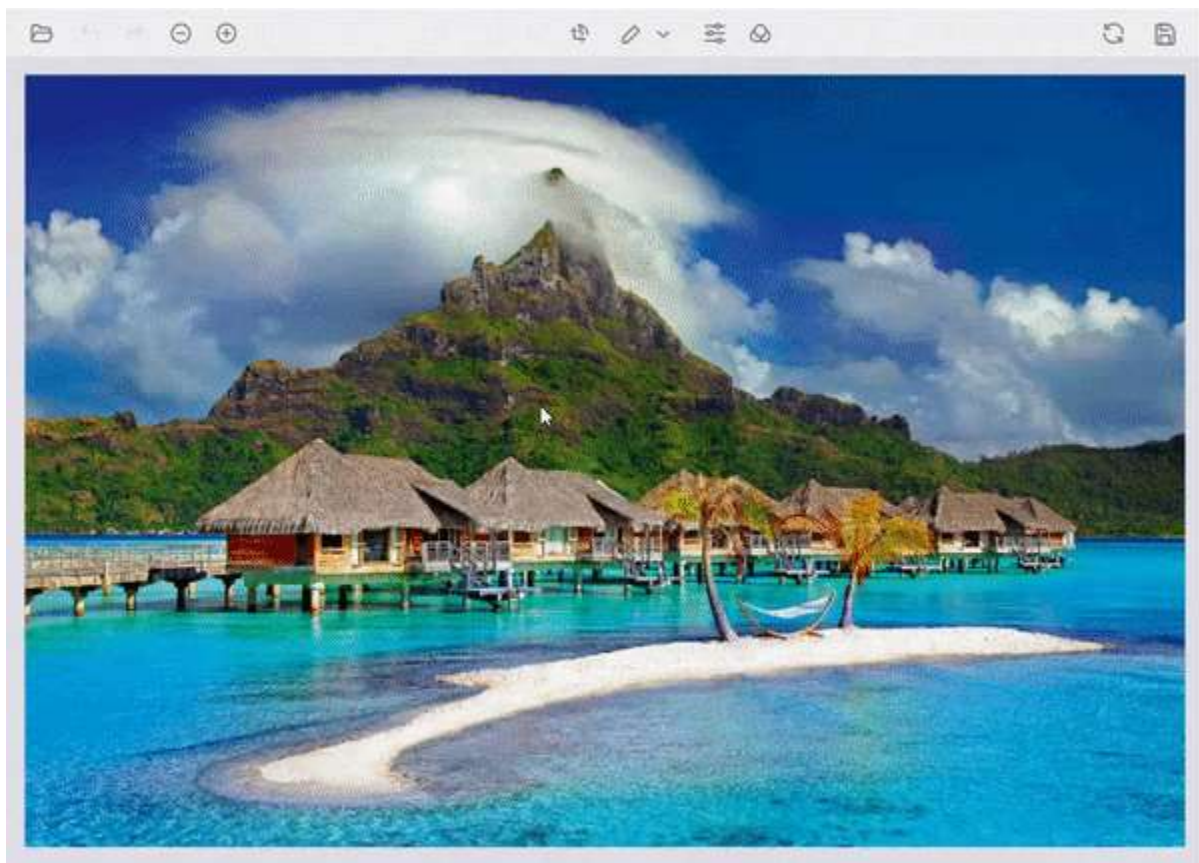
End-user capabilities in the EJ2 JavaScript Image Editor control

The following operations are available for end-users and the same is explained briefly in these sections.

Open an image

To open an image in the image editor, do the following steps.

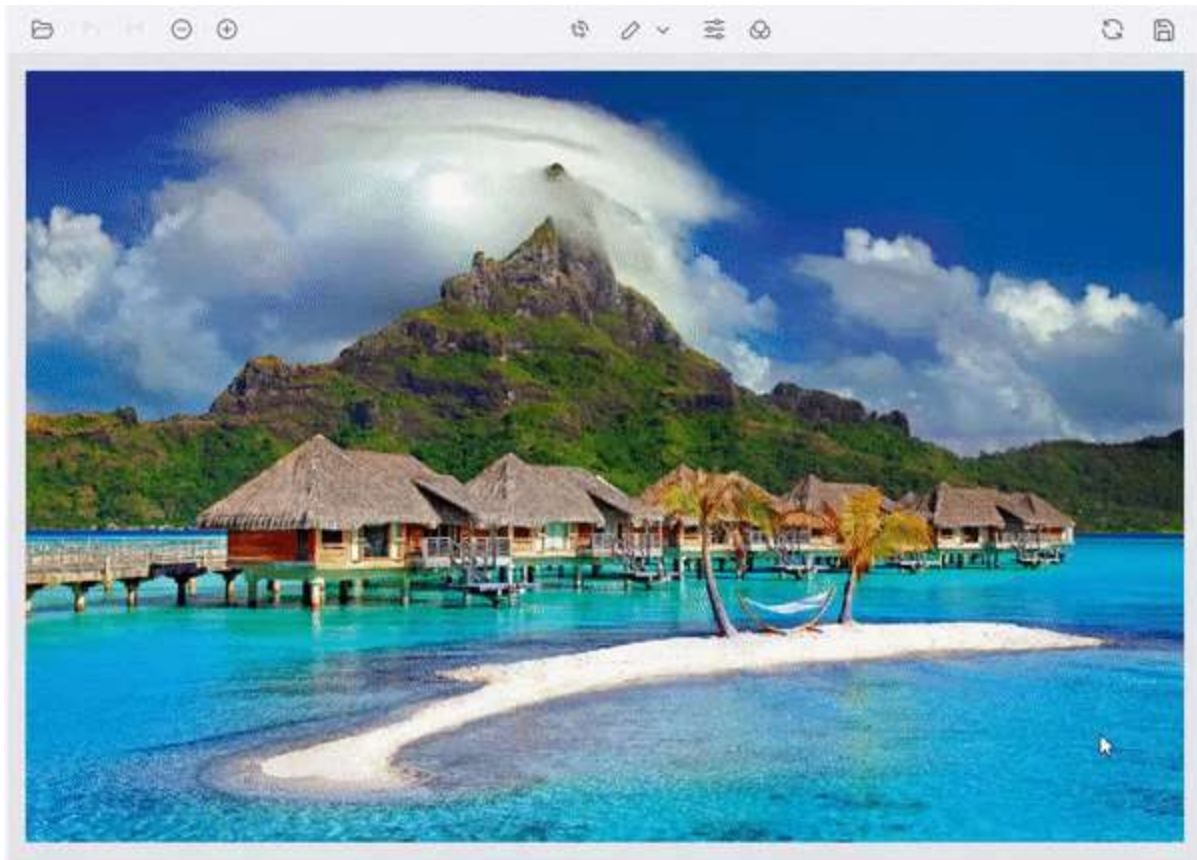
- Click the Open icon from the left side of the toolbar.
- The file explorer lists only JPEG, PNG, JPG format files.
- Select the image from the list of the images from the file explorer window.



Zooming

Image zooming can be performed in the following ways.

- Using toolbar.
- Using pinch zoom in touch enabled devices.
- Using mouse wheel.
- Using keyboard.



Using toolbar

To zoom in or out the image in the image editor, do the following steps.

- The Zoom In/ Out option only enabled after opening the image.

Using pinch

To zoom in or out the image in the image editor, do the following steps.

- Touch with two fingers to perform zooming.
- Zoom in and out controlled by touch gestures.

Using Mouse wheel

To zoom in or out the image in the image editor, do the following steps.

- Press the ctrl key and scroll the mouse wheel to perform zooming.
- The zoom in and out controlled by the mouse wheel.

Using keyboard

To zoom in or out the image in the image editor, do the following steps.

- Press the ctrl key with '+' button from the keyboard to zoom in an image.
- Press the ctrl key with '-' button from the keyboard to zoom out an image.

Panning

To pan an image in the image editor, do the following steps.

- Click on the image and do dragging to move or pan the image.
- Panning option will be enabled in the following two cases.
- If the selection is applied for cropping an image.
- If the image size exceeds the canvas size while zooming an image.



Cropping and image transformation

To crop an image in the image editor, do the following steps.

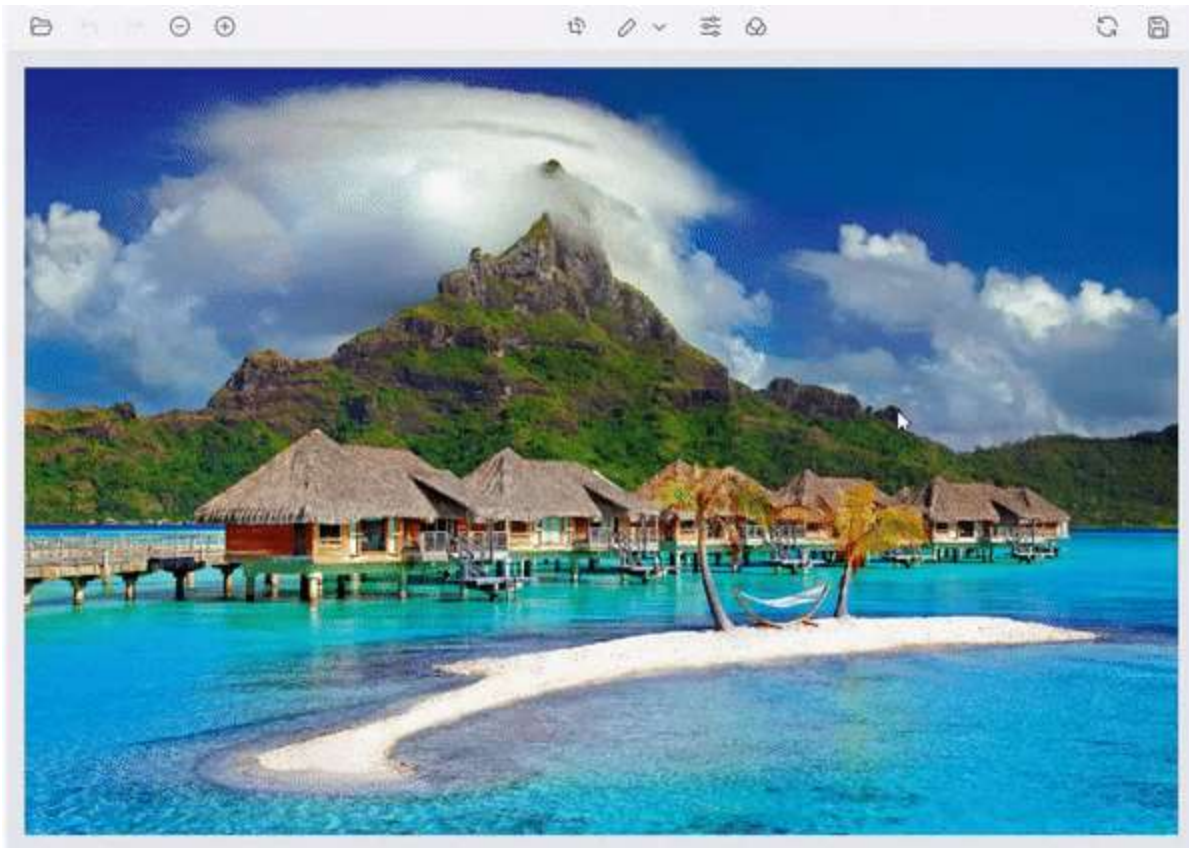
- Cropping can be performed based on the selection in an image editor.
- To perform selection, click the crop button in the toolbar which opens the contextual toolbar that shows crop selection options, rotate options, and flip options.
- Click the crop selection button and select the type of selection such as custom, circle, square, and ratio selection from the popup.
- Once selection is completed, do panning to move the image to get the cropped region.
- Utilize the rotate or flip buttons to execute the image transformation, including any inserted annotations.
- Once the cropping region is finalized in the image click the tick icon at the top right of the toolbar to crop the image.



Annotations

To add annotations to an image in the image editor, do the following steps.

- To add annotation, click the annotation button in the toolbar and select the type of annotations such as Line, Rectangle, Ellipse, Path, Arrow, Text, or Freehand drawing to be inserted to the image editor.
- Once the annotation is added to the image, that can be repositioned by clicking and dragging the annotations using mouse as well as resized by clicking and resizing the selection circle to be placed around the annotations.
- To rotate annotations, you can simply grab the circle located at the bottom of the annotation. The rotation can be applicable to all the annotations except text annotation.
- Customize the annotations by changing their color, stroke width, font family, and font size through the contextual toolbar. The contextual toolbar will be enabled whenever the annotations are selected.
- When annotations are selected in the Image Editor, the quick access toolbar becomes active, providing convenient access to various actions such as duplicating, deleting, or editing text associated with the selected annotation. This toolbar enables users to perform these common operations quickly and efficiently, streamlining their workflow and enhancing the overall editing experience.



Filtering and fine-tune

To perform fine-tuning on an image in the image editor, do the following steps.

- Click the fine-tune button which displays the list of fine-tuning available in the image editor.
- Click one of the fine-tune options from the list of options which shows a slider to adjust the corresponding filter.
- Click on the canvas or tick icon at the right corner of the toolbar in the image editor to apply the modifications.

To apply filters on an image in the image editor, do the following steps.

- Click the filter button which displays the list of filters available in the image editor.
- Click the filter from list of options to apply the corresponding filter to an image.
- Click on the canvas or tick icon at the right corner of the toolbar in the image editor to apply the modifications.

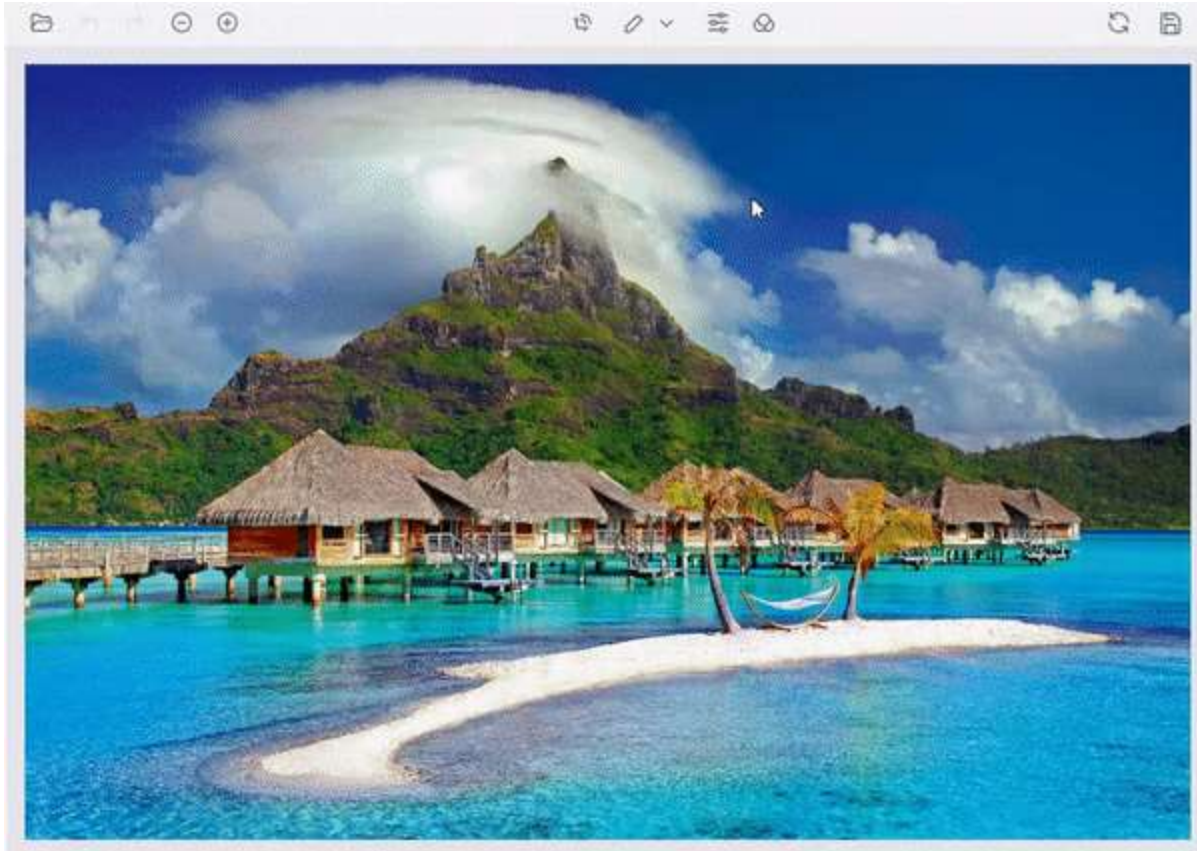
**Note:**

The Filters and Fine-tunes feature are not accessible within Safari due to compatibility limitations.

[Undo and redo the operations](#)

To undo and redo the actions performed in an image editor, do the following steps.

- The undo button will be enabled once the action is performed in an image editor.
- The redo button will be enabled once the undo action is performed in an image editor.
- Click the undo or redo button at the left side of the toolbar to perform undo and redo operation.
- Ctrl + Z and Ctrl + Y facilitates this process by allowing users to undo and redo actions, respectively.



Reset an image

To revert all the changes done in an image editor, do the following steps.

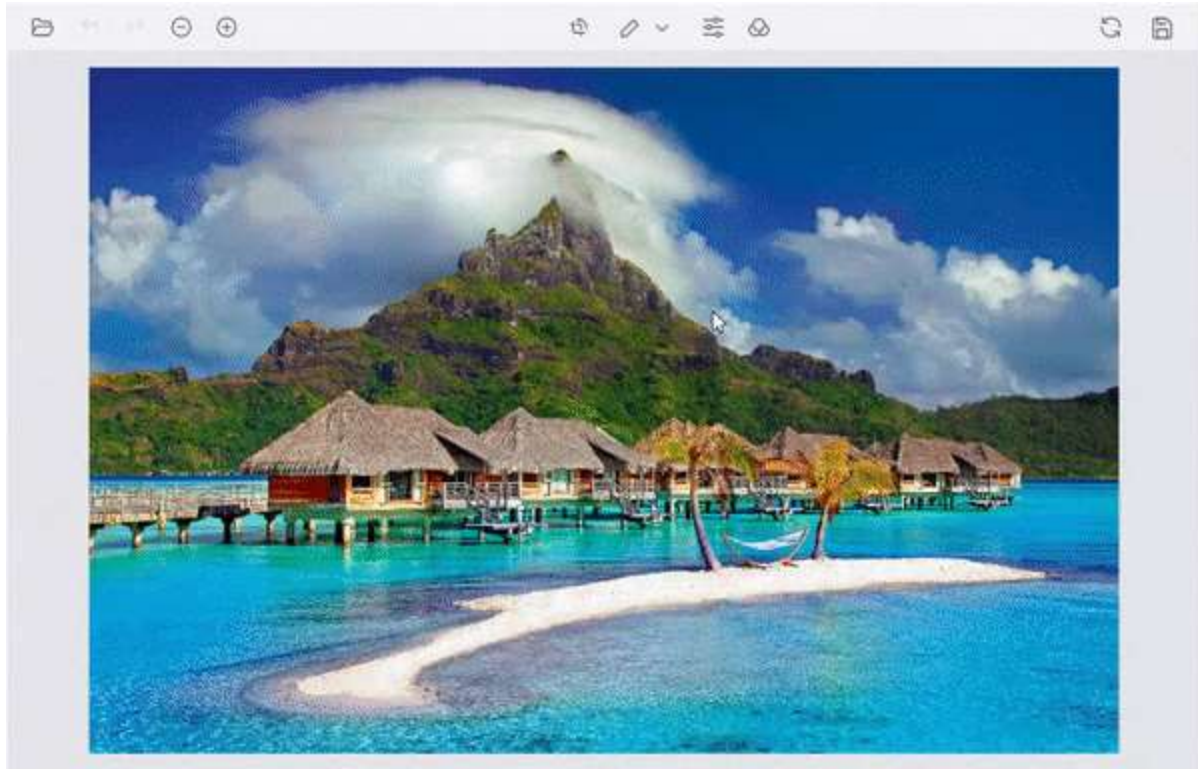
- Click the reset button which is located on the right side of the toolbar.
- This will revert all the changes performed in the image editor.



Export an image

To save the modified image in an image editor, do the following steps.

- Click the save button which is located on the right side of the toolbar.
- Ctrl + S facilitates this process by providing users with the ability to save the image.
- Select the type of file to be saved from the popup to save with current modification done in an image.



Open and save in the EJ2 JavaScript Image Editor control

To import an image into the canvas, it must first be converted into a blob object. The Uploader component can be used to facilitate the process of uploading an image from the user interface. Once the image has been uploaded, it can then be converted into a blob and drawn onto the canvas.

The `getImageData` method is used to get the image as `ImageData` and this can be loaded to our Image Editor control using the `open` method.

Open an image

The `open` method in the Image Editor control offers the capability to open an image by providing it in different formats. This method accepts various types of arguments, such as a base64-encoded string, raw image data, or a hosted/online URL. You can pass either the file name or the actual image data as an argument to the `open` method, and it will load the specified image into the Image Editor control. This flexibility allows you to work with images from different sources and formats, making it easier to integrate and manipulate images within the Image Editor control.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
```

```

        imageEditorObj.open('bee-eater.png');
    }
}
});
imageEditorObj.appendTo('#imageeditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
        </div>
      </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Supported image formats

The Image Editor control supports three common image formats: PNG, JPEG, and SVG. These formats allow you to work with a wide range of image files within the Image Editor.

When it comes to saving the edited image, the default file type is set as PNG. This means that when you save the edited image without specifying a different file type, it will be saved as a PNG file. However, it's important to note that the Image Editor typically provides options or methods to specify a different file type if desired. This allows you to save the edited image in formats other than the default PNG, such as JPEG or SVG, based on your specific requirements or preferences.

Save as image

The [export](#) method is used to save the modified image as an image, and it accepts a file name and file type as parameters. The file type parameter supports PNG, JPEG, and SVG and the default file type is PNG. It also saves an image by clicking the save button from the toolbar and the supported file types are PNG, JPEG, and SVG.

In the following example, the [export](#) method is used in the button click event.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click'}, '#btnClick');
document.getElementById('btnClick').onclick = (): void => {
  imageEditorObj.export("PNG", "Syncfusion"); // File type, file name
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
                <div>
                    <button id="btnClick" class="e-btn e-primary">Click</button>
                </div>
            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

File opened event

The [fileOpened](#) event is triggered in the Image Editor control after an image is successfully loaded. It provides the [openEventArgs](#) as the event argument, which contains two specific arguments:

fileName: This argument is a string that contains the file name of the opened image. It represents the name of the file that was selected or provided when loading the image into the Image Editor.

fileType: This argument is a string that contains the type of the opened image. It specifies the format or file type of the image that was loaded, such as PNG, JPEG, or SVG.

By accessing these arguments within the [fileOpened](#) event handler, you can retrieve information about the loaded image, such as its file name and file type. This can be useful for performing additional actions or implementing logic based on the specific image that was opened in the Image Editor control.

Saving event

The [saving](#) event is triggered in the Image Editor control when an image is being saved to the local disk. It provides the [SaveEventArgs](#) as the event argument, which includes the following specific arguments:

[fileName](#): This argument is a string that holds the file name of the saved image. It represents the name of the file that will be used when saving the image to the local disk.

[fileType](#): This argument is a string indicating the type or format of the saved image. It specifies the desired file type in which the image will be saved, such as PNG, JPEG, or SVG.

[cancel](#): This argument is a boolean value that can be set to true in order to cancel the saving action. By default, it is set to false, allowing the saving process to proceed. However, if you want to prevent the saving action from occurring, you can set Cancel to true within the event handler.

By accessing these arguments within the Saving event handler, you can retrieve information about the file name and file type of the image being saved. Additionally, you have the option to cancel the saving action if necessary.

Created event

The [created](#) event is triggered once the Image Editor control is created. This event serves as a notification that the component has been fully initialized and is ready to be used. It provides a convenient opportunity to render the Image Editor with a predefined set of initial settings, including the image, annotations, and transformations.

In the following example, the [created](#) event is used to load an image.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
```



```

<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Destroyed event

The [destroyed](#) event is triggered once the Image Editor control is destroyed or removed from the application. This event serves as a notification that the component and its associated resources have been successfully cleaned up and are no longer active.

Reset an image

The [reset](#) method in the Image Editor control provides the capability to undo all the changes made to an image and revert it back to its original state. This method is particularly useful when multiple adjustments, annotations, or transformations have been applied to an image and you want to start over with the original, unmodified version of the image.

By invoking the [reset](#) method, any modifications or edits made to the image will be undone, and the image will be restored to its initial state. This allows you to easily discard any changes and begin again with the fresh, unaltered image.

Selection cropping in the EJ2 JavaScript Image Editor control

The cropping feature in the Image Editor allows you to select and crop specific regions of an image. It offers different selection options, including custom shapes, squares, circles, and various aspect ratios such as 2:3, 3:2, 3:4, 4:3, 4:5, 5:4, 5:7, 7:5, 9:16, and 16:9.

To perform a selection, you can use the [select](#) method, which allows you to define the desired selection area within the image. Once the selection is made, you can then use the [crop](#) method to crop the image based on the selected region. This enables you to extract and focus on specific parts of the image while discarding the rest.

Insert custom / square / circle region

The [select](#) method allows to perform selection based on the type of selection. Here, the [select](#) method is used to perform the selection as custom, circle, or square. The selection region can also be customized using the select method based on the parameters below.

type - Specify the type of selection

startX - Specify the x-coordinate of the selection region's starting point

startY - Specify the y-coordinate of the selection region's starting point

width - Specify the width of the selection region

height - Specify the height of the selection region

Here is an example of square selection using the [select](#) method.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click', '#btnClick'});
document.getElementById('btnClick').onclick = (): void => {
  imageEditorObj.select("Square");
}
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
      </div>
      <div>
        <button id="btnClick" class="e-btn e-primary">Click</button>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Insert selection based on aspect ratio

The [select](#) method is used to perform the selection with the various aspect ratios such as 2:3, 3:2, 3:4, 4:3, 4:5, 5:4, 5:7, 7:5, 9:16, and 16:9. The selection region can also be customized using the SelectAsync method based on the parameters below.

- type - Specify the type of selection
- startX - Specify the x-coordinate of the selection region's starting point
- startY - Specify the y-coordinate of the selection region's starting point

Here is an example of ratio selection using the [select](#) method.

In the following example, the [select](#) method is used in the button click to the ratio selection.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click'}, '#btnClick');
document.getElementById('btnClick').onclick = (): void => {
  imageEditorObj.select("16:9");
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
            </div>
            <div>
                <button id="btnClick" class="e-btn e-primary">Click</button>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Resize selections

The selection region can be changed programmatically by using [selectionChanging](#) event. This event is activated during resizing the selection using mouse, and it allows for alterations to the selection region by adjusting the specified properties.

The [SelectionChangeEventArgs](#) is used in this event to customize the selection and it has the following parameters.

SelectionChangeEventArgs.cction - The type of action such as inserting or resizing

SelectionChangeEventArgs.cancel - Specifies to cancel the selection.

SelectionChangeEventArgs.currentSelectionPoint - Represents all the details of the selection including its type, position, width, and height after the current action as CropSelectionSettings.

SelectionChangeEventArgs.previousSelectionPoint - Represents all the details of the selection including its type, position, width, and height before this current action as CropSelectionSettings

Here is an example of changing the selection region using the [SelectionChangeEventArgs](#) event.

Crop an image

The [crop](#) method allows cropping based on the selected region. Here is an example of cropping the selection region using the [crop](#) method.

Here is an example of circle cropping using the [select](#) and [crop](#) method.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click'}, '#btnClick');
document.getElementById('btnClick').onclick = (): void => {
  imageEditorObj.select("Circle");
  imageEditorObj.crop();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
                <div>
                    <button id="btnClick" class="e-btn e-primary">Click</button>
                </div>
            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cropping event

The [cropping](#) event is triggered when performing cropping on the image. This event is passed an object that contains information about the cropping event, such as the start and end point of the selection region. And this event uses [CropEventArgs](#) to handle the cropping action in the image.

The parameter available in the [cropping](#) event is,

- CropEventArgs.startPoint – The x and y coordinates of a start point as [Point](#) of the selection region.
- CropEventArgs.endPoint - The x and y coordinates of an end point as [Point](#) of the selection region.
- CropEventArgs.cancel - To cancel the cropping action.

Annotation in the EJ2 JavaScript Image Editor control

The Image Editor allows adding annotations to the image, including text, freehand drawings, and shapes like rectangles, ellipses, arrows, paths, and lines. This gives the flexibility to mark up the image with

notes, sketches, and other visual elements as needed. These annotation tools can help to communicate and share ideas more effectively.

Text annotation

The text annotation feature in the Image Editor provides the capability to add and customize labels, captions, and other text elements directly onto the image. With this feature, you can easily insert text at specific locations within the image and customize various aspects of the text to meet your requirements.

You have control over the customization options including text content, font family, font style and font size for the text annotation.

Add a text

The [drawText](#) method in the Image Editor allows you to insert a text annotation into the image with specific customization options. This method accepts the following parameters:

- x - Specifies the x-coordinate of the text, determining its horizontal position within the image.
- y - Specifies the y-coordinate of the text, determining its vertical position within the image.
- text - Specifies the actual text content to be added to the image.
- fontFamily - Specifies the font family of the text, allowing you to choose a specific typeface or style for the text.
- fontSize - Specifies the font size of the text, determining its relative size within the image.
- bold - Specifies whether the text should be displayed in bold style. Set to true for bold text, and false for regular text.
- italic - Specifies whether the text should be displayed in italic style. Set to true for italic text, and false for regular text.
- color - Specifies the font color of the text, allowing you to define the desired color using appropriate color values or names.

By utilizing the [drawText](#) method with these parameters, you can precisely position and customize text annotations within the image. This provides the flexibility to add labels, captions, or other text elements with specific font styles, sizes, and colors, enhancing the visual presentation and clarity of the image.

Here is an example of adding a text in a button click using [drawText](#) method.

In the following example, you can using the [drawText](#) method in the button click event.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
```



```

imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click'}, '#btnClick');
document.getElementById('btnClick').onclick = (): void => {
    let dimension: any = imageEditorObj.getImageDimension();
    imageEditorObj.drawText(dimension.x, dimension.y, 'Syncfusion');
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
      </div>
      <div>
        <button id="btnClick" class="e-btn e-primary">Click</button>
      </div>
    </div>
  </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiline text

The [drawText](#) method in the Image Editor control is commonly used to insert text annotations into an image. If the provided text parameter contains a newline character (\n), the text will be automatically split into multiple lines, with each line appearing on a separate line in the annotation.

In the following example, you can using the [drawText](#) method in the button click event.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click', '#btnClick'});
document.getElementById('btnClick').onclick = (): void => {
  let dimension: any = imageEditorObj.getImageDimension();
  imageEditorObj.drawText(dimension.x, dimension.y, 'Enter\nText');
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
                <div>
                    <button id="btnClick" class="e-btn e-primary">Click</button>
                </div>
            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Delete a text

The [deleteShape](#) method in the Image Editor allows you to remove a text annotation from the image editor. To use this method, you need to pass the [shapeId](#) of the annotation as a parameter.

The [shapeId](#) is a unique identifier assigned to each text annotation within the image editor. It serves as a reference to a specific annotation, enabling targeted deletion of the desired text element. By specifying the [shapeId](#) associated with the text annotation you want to remove, you can effectively delete it from the image editor.

To retrieve the inserted text annotations, you can utilize the [getShapeSetting](#) method, which provides a collection of annotations represented by [ShapeSettings](#). This method allows you to access and work with the annotations that have been inserted into the image.

In the following example, the [deleteShape](#) method is used in the button click event.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
document.getElementById('btnClick').onclick = (): void => {
  let dimension: any = imageEditorObj.getImageDimension();
  imageEditorObj.drawText(dimension.x, dimension.y, 'Enter\nText');
}
let delbutton: Button = new Button({cssClass: `e-primary`,
content: 'Delete Text'}, '#delClick');
document.getElementById('delClick').onclick = (): void => {
  imageEditorObj.deleteShape('shape_1');
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
  editor/styles/material.css" rel="stylesheet">
```

```

<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
            <div>
                <button id='btnClick' class='e-btn e-primary'
>Click</button>
                <button id='delClick' class='e-btn e-primary'
>Click</button>
            </div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize font family and text color

The [shapeChanging](#) event in the Image Editor control is triggered when a text annotation is being modified or changed through the toolbar interaction. This event provides an opportunity to make alterations to the text's color and font family by adjusting the relevant properties.

By leveraging the [shapeChanging](#) event, you can enhance the customization options for text annotations and provide a more tailored and interactive experience within the Image Editor control.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    shapeChanging: (args: shapeChanging) => {
        if (args.currentShapeSettings.type === 'Text') {
            args.currentShapeSettings.color = 'red';
        }
    },
    created: () => {

```

```

    if (Browser.isDevice) {
        imageEditorObj.open('bee-eater.png');
    } else {
        imageEditorObj.open('bee-eater.png');
    }
}
});
imageEditorObj.appendTo('#imageeditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
        </div>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add Additional font family

The [fontFamily](#) property in the Image Editor control provides the flexibility to incorporate supplementary font families, expanding your options for text styling and ensuring a broader range of fonts can be utilized within your design or content. The font value will be determined by the 'id' property.

By leveraging the [fontFamily](#) property, you can elevate the scope of customization for text annotations, enriching the user experience within the Image Editor control. This enhancement offers a more personalized and dynamic interaction, empowering users to tailor their text styles for a truly engaging editing experience.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  fontFamily: { default: 'Arial', items: [{id: 'arial', text: 'Arial'},
{id: 'brush script mt', text: 'Brush Script MT'},
{id: 'papyrus', text: 'Papyrus'}, {id: 'times new roman', text: 'Times
New Roman'}, {id: 'courier new', text: 'Courier New'}] },
  created: () => {
    imageEditorObj.open('bee-eater.png');
  }
});
imageEditorObj.appendTo('#imageeditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
            </div>
        </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Freehand drawing

The Freehand Draw annotation tool in the Image Editor control is a versatile feature that allows users to draw and sketch directly on the image using mouse or touch input. This tool provides a flexible and creative way to add freehand drawings or annotations to the image.

The [freehandDraw](#) method is used to enable or disable the freehand drawing option in the Image Editor control.

Here is an example of using the [freeHandDraw](#) method in a button click event.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    toolbar: [],
    created: () => {
        if (Browser.isDevice) {
            imageEditorObj.open('bee-eater.png');
        }
    }
});

```



```

    } else {
        imageEditorObj.open('bee-eater.png');
    }
}

});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click'}, '#btnClick');
document.getElementById('btnClick').onclick = (): void => {
    imageEditorObj.freeHandDraw(true);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
      </div>
      <div>
        <button id="btnClick" class="e-btn e-primary">Click</button>

```

```

        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adjust the stroke width and color

The [shapeChanging](#) event in the Image Editor control is triggered when a freehand annotation is being modified or changed through the toolbar interaction. This event provides an opportunity to make alterations to the freehand annotation's color and stroke width by adjusting the relevant properties.

By leveraging the [shapeChanging](#) event, you can enhance the customization options for freehand annotations and provide a more tailored and interactive experience within the Image Editor control.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    shapeChanging: (args: shapeChanging) => {
        if (args.currentShapeSettings.type === 'FreehandDraw') {
            args.currentShapeSettings.color = 'red',
            args.currentShapeSettings.strokeWidth = 10
        }
    },
    created: () => {
        if (Browser.isDevice) {
            imageEditorObj.open('bee-eater.png');
        } else {
            imageEditorObj.open('bee-eater.png');
        }
    }
});
imageEditorObj.appendTo('#imageeditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
                <div>
                    <button id="btnClick" class="e-btn e-primary">Click</button>
                </div>
            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Delete a freehand drawing

The [deleteShape](#) method in the Image Editor allows you to remove a freehand annotation from the image editor. To use this method, you need to pass the [shapeld](#) of the annotation as a parameter.

The [shapeld](#) is a unique identifier assigned to each freehand annotation within the image editor. It serves as a reference to a specific annotation, enabling targeted deletion of the desired annotation. By specifying the [shapeld](#) associated with the freehand annotation you want to remove, you can effectively delete it from the image editor.

To retrieve the inserted freehand annotations, you can utilize the [getShapeSetting](#) method, which provides a collection of annotations represented by [ShapeSettings](#). This method allows you to access and work with the annotations that have been inserted into the image.

In the following example, the [deleteShape](#) method is used in the button click event.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    toolbar: [],
    created: () => {
        if (Browser.isDevice) {
            imageEditorObj.open('bee-eater.png');
        } else {
            imageEditorObj.open('bee-eater.png');
        }
    }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
document.getElementById('btnClick').onclick = (): void => {
    imageEditorObj.freeHandDraw(true);
}
document.getElementById('delClick').onclick = (): void => {
    imageEditorObj.deleteShape('pen_1');
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
            <div>
                <button id='btnClick' class='e-btn e-primary'
>Enable</button>
                <button id='delClick' class='e-btn e-primary'
>Delete</button>
            </div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Shape annotation

The Image Editor control provides the ability to add shape annotations to an image. These shape annotations include rectangles, ellipses, arrows, paths, and lines, allowing you to highlight, emphasize, or mark specific areas or elements within the image.

Add a rectangle / ellipse / line / arrow / path

The [drawRectangle](#) method is used to insert a rectangle to the Image Editor control. Rectangle annotations are valuable tools for highlighting, emphasizing, or marking specific areas of an image to draw attention or provide additional context.

The [drawRectangle](#) method in the Image Editor control takes seven parameters to define the properties of the rectangle annotation:

- x: Specifies the x-coordinate of the top-left corner of the rectangle.
- y: Specifies the y-coordinate of the top-left corner of the rectangle.
- width: Specifies the width of the rectangle.
- height: Specifies the height of the rectangle.
- strokeWidth: Specifies the stroke width of the rectangle's border.
- strokeColor: Specifies the stroke color of the rectangle's border.

- `fillColor`: Specifies the fill color of the rectangle.

The [drawEllipse](#) method is used to insert a ellipse to the Image Editor control. Ellipse annotations are valuable for highlighting, emphasizing, or marking specific areas of an image.

The [drawEllipse](#) method in the Image Editor control takes seven parameters to define the properties of the ellipse annotation:

- `x`: Specifies the x-coordinate of the center of the ellipse.
- `y`: Specifies the y-coordinate of the center of the ellipse.
- `radiusX`: Specifies the horizontal radius (radiusX) of the ellipse.
- `radiusY`: Specifies the vertical radius (radiusY) of the ellipse.
- `strokeWidth`: Specifies the width of the ellipse's stroke (border).
- `strokeColor`: Specifies the color of the ellipse's stroke (border).
- `fillColor`: Specifies the fill color of the ellipse.

The [drawLine](#) method is used to insert a line to the Image Editor control. Line annotations are valuable for highlighting, emphasizing, or marking specific areas of an image.

The [drawLine](#) method in the Image Editor control takes seven parameters to define the properties of the ellipse annotation:

- `startX` - Specifies the x-coordinate of the start point.
- `startY` - Specifies the y-coordinate of the start point.
- `endX` - Specifies the x-coordinate of the end point.
- `endY` - Specifies the y-coordinate of the end point.
- `strokeWidth` - Specifies the stroke width of the line.
- `strokeColor` - Specifies the stroke color of the line.

The [drawArrow](#) method is used to insert a arrow to the Image Editor control. Arrow annotations are valuable for highlighting, emphasizing, or marking specific areas of an image.

The [drawArrow](#) method in the Image Editor control takes seven parameters to define the properties of the ellipse annotation:

- `startX` - Specifies the x-coordinate of the start point.
- `startY` - Specifies the y-coordinate of the start point.
- `endX` - Specifies the x-coordinate of the end point.
- `endY` - Specifies the y-coordinate of the end point.
- `strokeWidth` - Specifies the stroke width of the arrow.
- `strokeColor` - Specifies the stroke color of the arrow.
- `arrowStart` - Specifies the arrowhead as `ImageEditorArrowHeadType` at the start of arrow.
- `arrowEnd` - Specifies the arrowhead as `ImageEditorArrowHeadType` at the end of the arrow.

The [drawPath](#) method is used to insert a path to the Image Editor control. Path annotations are valuable for highlighting, emphasizing, or marking specific areas of an image.

The [drawPath](#) method in the Image Editor control takes three parameters to define the properties of the ellipse annotation:

- **points** - Specifies collection of x and y coordinates as ImageEditorPoint to draw a path.
- **strokeWidth** - Specifies the stroke width of the path.
- **strokeColor** - Specifies the stroke color of the path.

Here is an example of inserting rectangle, ellipse, arrow, path, and line in a button click event.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click'}, '#btnClick');
document.getElementById('rectangleClick').onclick = (): void => {
  let dimension: any = imageEditorObj.getImageDimension();
  imageEditorObj.drawRectangle(dimension.x, dimension.y);
}
document.getElementById('ellipseClick').onclick = (): void => {
  let dimension: any = imageEditorObj.getImageDimension();
  imageEditorObj.drawEllipse(dimension.x, dimension.y);
}
document.getElementById('lineClick').onclick = (): void => {
  let dimension: any = imageEditorObj.getImageDimension();
  imageEditorObj.drawLine(dimension.x, dimension.y);
}
document.getElementById('arrowClick').onclick = (): void => {
  let dimension: any = imageEditorObj.getImageDimension();
  imageEditorObj.drawArrow(dimension.x, dimension.y+10,
dimension.x+50, dimension.y+10, 10);
}
document.getElementById('pathClick').onclick = (): void => {
  let dimension: any = imageEditorObj.getImageDimension();
  imageEditorObj.drawPath([{x: dimension.x, y: dimension.y}, {x:
dimension.x+50, y: dimension.y+50}, {x: dimension.x+20, y: dimension.y+50}],
8);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                <div>
                    <button id='rectangleClick' class='e-btn e-primary'
>Rectangle</button>
                    <button id='ellipseClick' class='e-btn e-primary'
>Ellipse</button>
                    <button id='lineClick' class='e-btn e-primary'
>Line</button>
                    <button id='arrowClick' class='e-btn e-primary'
>Arrow</button>
                    <button id='pathClick' class='e-btn e-primary'
>Path</button>
                </div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Delete a shape

The [deleteShape](#) method in the Image Editor allows you to remove a shape annotation from the image editor. To use this method, you need to pass the [shapeld](#) of the annotation as a parameter.

The [shapeld](#) is a unique identifier assigned to each shape annotation within the image editor. It serves as a reference to a specific annotation, enabling targeted deletion of the desired annotation. By specifying the [shapeld](#) associated with the shape annotation you want to remove, you can effectively delete it from the image editor.

To retrieve the inserted shape annotations, you can utilize the [getShapeSetting](#) method, which provides a collection of annotations represented by [ShapeSettings](#). This method allows you to access and work with the annotations that have been inserted into the image.

Here is an example of deleting rectangle, ellipse, arrow, path, and line in a button click event.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let id: string;
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: ['Annotate', 'Line', 'Rectangle', 'Ellipse', 'Circle', 'Arrow',
    'Path'],
  showQuickAccessToolbar: false,
  shapeChanging: (args: shapeChanging) => {
    if (args.action === 'select') {
      id = args.currentShapeSettings.id;
    }
  },
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
document.getElementById('btnClick').onclick = (): void => {
  imageEditorObj.deleteShape(id);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
            <div>
                <button id="btnClick" class="e-btn e-primary">Click</button>
            </div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Image annotation

The image annotation feature in the Image Editor provides the capability to add and customize images directly onto the image. With this feature, you can easily insert image or icons at specific locations within the image and customize various aspects of the image to meet your requirements. You have control over the customization options including rotate, flip, transparency for the image annotation.

Add an image annotation

The [drawImage](#) method serves the purpose of inserting an image into the Image Editor control, allowing for image annotations to be added. These image annotations can be used for various purposes, such as adding logos, watermarks, or decorative elements to the image.

The [drawImage](#) method in the Image Editor control takes six parameters to define the properties of the image annotation:

- data: Specified the image data or url of the image to be inserted.
- x: Specifies the x-coordinate of the top-left corner of the image.
- y: Specifies the y-coordinate of the top-left corner of the image.
- width: Specifies the width of the image.
- height: Specifies the height of the image.
- isAspectRatio: Specifies whether the image is rendered with aspect ratio or not.

In the following example, you can use the [drawImage](#) method in the button click event.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    toolbar: [],
    created: () => {
        imageEditorObj.open('bee-eater.png');
    }
});
imageEditorObj.appendTo('#imageeditor');
document.getElementById('btnClick').onclick = (): void => {
    let dimension: any = imageEditorObj.getImageDimension();
    imageEditorObj.drawImage('flower.png', dimension.x, dimension.y,
100, 80, true, 0);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
            <div>
                <button id='btnClick' class='e-btn e-primary' >Add
Image</button>
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Transform in the EJ2 JavaScript Image Editor control

The Image Editor provides a range of transformation options for manipulating both the image and its annotations. These options include rotation, flipping, zooming, and panning. These transformations offer flexibility in adjusting the image and enhancing its visual appearance.

Rotate an image

The Image Editor allows to rotate the image and its annotations by a specific number of degrees clockwise or anti-clockwise using [rotate](#) method. This method takes a single parameter: the angle of rotation in degrees. A positive value will rotate the image clockwise, while a negative value will rotate it anti-clockwise.

Here is an example of rotating an image in a button click event.

In the following example, the [rotate](#) method is used to rotate the image.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    toolbar: [],
    created: () => {
        if (Browser.isDevice) {
            imageEditorObj.open('bee-eater.png');
        } else {
            imageEditorObj.open('bee-eater.png');
        }
    }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click'}, '#btnClick');
document.getElementById('btnClick').onclick = (): void => {
    imageEditorObj.rotate(90);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
                <div>
                    <button id="btnClick" class="e-btn e-primary">Click</button>
                </div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Flip an image

The Image Editor provides the [flip](#) method, which allows you to flip both the image and its annotations either horizontally or vertically. This method takes a single parameter of type [Direction](#), which specifies the direction in which the flip operation should be applied.

The [Direction](#) parameter accepts two values: 'Horizontal' and 'Vertical'. When you choose 'Horizontal', the image and annotations will be flipped along the horizontal axis, resulting in a mirror effect. On the other hand, selecting 'Vertical' will flip them along the vertical axis, producing a vertical mirror effect.

Here is an example of flipping an image in a button click event.

In the following example, the [flip](#) method is used to flip the image.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    toolbar: [],
    created: () => {
        if (Browser.isDevice) {
            imageEditorObj.open('bee-eater.png');
        } else {
            imageEditorObj.open('bee-eater.png');
        }
    }
});

```

```

imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click'}, '#btnClick');
document.getElementById('btnClick').onclick = (): void => {
    imageEditorObj.flip("Horizontal"); // Horizontal flip
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
        </div>
        <div>
          <button id="btnClick" class="e-btn e-primary">Click</button>
        </div>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Straighten an image

The straightening feature in an Image Editor allows users to adjust an image by rotating it clockwise or counter clockwise. The rotating degree value should be within the range of -45 to +45 degrees for accurate straightening. Positive values indicate clockwise rotation, while negative values indicate counter clockwise rotation. The Image Editor control includes a [straightenImage](#) method, which allows you to adjust the degree of an image. This method takes one parameter that define how the straightening should be carried out:

- **degree:** Specifies the amount of rotation for straightening the image. Positive values indicate clockwise rotation, while negative values indicate counterclockwise rotation.

Here is an example of straightening the image.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    created: () => {
        imageEditorObj.open('bee-eater.png');
    }
});
imageEditorObj.appendTo('#imageeditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Zoom in or out an image

The Image Editor allows to magnify an image using the [zoom](#) method. This method allows one to zoom in and out of the image and provides a more detailed view of the image's hidden areas. This method takes two parameters to perform zooming.

zoomFactor - Specifies a value to controlling the level of magnification applied to the image.

zoomPoint - Specifies x and y coordinates of a point as [Point](#) on image to perform zooming.

Here is an example of zooming an image in a button click event.

In the following example, you can using the [zoom](#) method in the button click event.

INDEX.TS

```

import { ImageEditor, ZoomSettingsModel } from '@syncfusion/ej2-image-
editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    toolbar: [],

```

```

zoomSettings: {maxZoomFactor: 30},
zoomLevel: 1,
created: () => {
    if (Browser.isDevice) {
        imageEditorObj.open('bee-eater.png');
    } else {
        imageEditorObj.open('bee-eater.png');
    }
}
});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click'}, '#btnClick');
document.getElementById('btnClick').onclick = (): void => {
    if (imageEditorObj.zoomLevel < 1) {
        imageEditorObj.zoomLevel += 0.1;
    }
    else {
        imageEditorObj.zoomLevel += 1;
    }
    if (imageEditorObj.zoomLevel >
imageEditorObj.zoomSettings.maxZoomFactor) {
        imageEditorObj.zoomLevel =
imageEditorObj.zoomSettings.maxZoomFactor;
    }
    imageEditorObj.zoom(imageEditorObj.zoomLevel); // Zoom in
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">

```

```

<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
            </div>
            <div>
                <button id="btnClick" class="e-btn e-primary">Click</button>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To perform the Zoom out the image, In toolbar, you can clicking the Zoom out button in toolbar.

In the following example, you can using the [zoom](#) method in the button click event.

INDEX.TS

```

import { ImageEditor, ZoomSettingsModel } from '@syncfusion/ej2-image-
editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    toolbar: [],
    zoomSettings: {minZoomFactor: 0.1},
    zoomLevel: 1,
    created: () => {
        if (Browser.isDevice) {
            imageEditorObj.open('bee-eater.png');
        } else {
            imageEditorObj.open('bee-eater.png');
        }
    }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click'}, '#btnClick');

```

```

document.getElementById('btnClick').onclick = (): void => {
    if (imageEditorObj.zoomLevel < 1) {
        imageEditorObj.zoomLevel -= 0.1;
    }
    else {
        imageEditorObj.zoomLevel -= 1;
    }
    if (imageEditorObj.zoomLevel <
imageEditorObj.zoomSettings.minZoomFactor) {
        imageEditorObj.zoomLevel =
imageEditorObj.zoomSettings.minZoomFactor;
    }
    imageEditorObj.zoom(imageEditorObj.zoomLevel); // Zoom out
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                <div>

```

```

        <div>
            <button id="btnClick" class="e-btn e-primary">Zoom
IN</button>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Maximum and Minimum zoom level

The [maxZoomFactor](#) property is a useful feature in the Image Editor that allows you to define the maximum level of zoom permitted for an image. This property sets a limit on how much the image can be magnified, preventing excessive zooming that may result in a loss of image quality or visibility.

By default, the [maxZoomFactor](#) value is set to 10, meaning that the image can be zoomed in up to 10 times its original size. This ensures that the zooming functionality remains within reasonable bounds and maintains the integrity of the image.

The [minZoomFactor](#) property allows you to specify the minimum level of zoom that is allowed for an image. By setting this property, you can prevent the image from being zoomed out beyond a certain point, ensuring that it remains visible and usable even at the smallest zoom level.

By default, the [minZoomFactor](#) value is set to 0.1, meaning that the image can be zoomed out up to 10 times its original size.

Here is an example of specifying [minZoomFactor](#) and [maxZoomFactor](#) property in [zoomSettings](#) options in an image editor.

Panning event

The [panning](#) event is activated when the user begins dragging the image within the canvas. This event provide an opportunity to perform specific actions, like adjusting the position of an image, in response to the gesture of panning. And these event uses [panEventArgs](#) to handle the panning action when the user starts dragging the image.

The parameter available in the [panEventArgs](#) events are,

- [PanEventArgs.startPoint](#) - The x and y coordinates as [ImageEditorPoint](#) for the start point.
- [PanEventArgs.endpoint](#) - The x and y coordinates as [ImageEditorPoint](#) for the end point.
- [PanEventArgs.cancel](#) – Specifies the boolean value to cancel the panning action.

In the following example, you can use the [pan](#) method in the button click event.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition

```

```

let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click'}, '#btnClick');
document.getElementById('btnClick').onclick = (): void => {
  imageEditorObj.zoom(2); // Zoom in
  imageEditorObj.pan(true);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
      </div>
      <div>
        <button id="btnClick" class="e-btn e-primary">Click</button>
      </div>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Zooming event

The [zooming](#) event is triggered when performing zooming the image. This event can be used to perform certain actions, such as updating the position of the image. This event is passed an object that contains information about the zooming event, such as the amount of zooming performed. And this event uses [ZoomEventArgs](#) to handle the zooming action in the image.

The parameter available in the [zooming](#) event is,

- ZoomEventArgs.zoomPoint - The x and y coordinates as [Point](#) for the zoom point.
- ZoomEventArgs.previousZoomFactor - The previous zoom factor applied in the image editor.
- ZoomEventArgs.currentZoomFactor - The current zoom factor to be applied in the image editor.
- ZoomEventArgs.cancel – Specify a boolean value to cancel the zooming action.
- ZoomEventArgs.zoomTrigger - The type of zooming performed in the image editor.

Rotating event

The [rotating](#) event is triggered when performing rotating the image. This event is passed an object that contains information about the rotating event, such as the amount of rotation performed. And this event uses [RotateEventArgs](#) to handle the rotating action in the image.

The parameter available in the [rotating](#) event is,

- RotateEventArgs.previousDegree: The degree of rotation before the recent rotation action was applied in the Image Editor.
- RotateEventArgs.currentDegree: The current degree of rotation after the rotation action has been performed in the Image Editor.
- RotateEventArgs.cancel - Specifies a boolean value to cancel the rotating action.

Flipping event

The [flipping](#) event is triggered when performing flipping the image. This event is passed an object that contains information about the flipping event, such as the amount of flip performed. And this event uses [FlipEventArgs](#) to handle the flipping action in the image.

The parameter available in the [flipping](#) event is,

- FlipEventArgs.direction - The flip direction as [Direction](#) to be applied in the image editor.
- FlipEventArgs.cancel - Specifies a boolean value to cancel the flip action.

Toolbar in the EJ2 JavaScript Image Editor control

The toolbars in the Image Editor are a key component for interacting with and editing images. They provide a range of tools and options that can be customized to suit the needs and preferences. Add or remove items from the toolbar to create a personalized set of tools, or they can even create their own custom toolbar from scratch. This flexibility and customization allow them to create a unique image editing experience that is tailored to their specific needs and workflow.

In the Image Editor, the [toolbar](#) provides the ability to customize the toolbar by adding or removing items, as well as defining a completely custom toolbar. This feature is valuable for creating a personalized image editing experience that aligns with specific requirements and workflows.

Built-in toolbar items

Specifies the toolbar items to perform UI interactions. Refer to the built-in toolbar items for the default value.

- Crop
- Transform
- Annotate
- ZoomIn
- ZoomOut
- Open
- Reset
- Save
- Pan

Add a custom toolbar item

The [toolbar](#) property in the Image Editor allows to add or remove toolbar items to include only the tools they frequently use, streamlining the editing process and reducing clutter.

Here is an example of adding custom toolbar items to rotate and flip transformation using [toolbar](#) property.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  toolbar: ['Crop', 'CustomSelection', 'SquareSelection', 'ZoomIn',
'ZoomOut', {text: 'Rotate'}],
  width: '550px',
  height: '350px',
  toolbarItemClicked: (args: ToolbarClickEventArgs) => {
    if (args.item.text === 'Rotate') {
      imageEditorObj.rotate(90);
    }
  },
},
```



```

created: () => {
  if (Browser.isDevice) {
    imageEditorObj.open('bee-eater.png');
  } else {
    imageEditorObj.open('bee-eater.png');
  }
}
});
imageEditorObj.appendTo('#imageeditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
        </div>
      </div>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide a toolbar

The [toolbar](#) property controls the visibility of the toolbar in the Image Editor. When the [toolbar](#) is set to an empty list, the toolbar is hidden. Conversely, if the [toolbar](#) contains a list of items, the toolbar is shown, displaying the specified items. This feature provides flexibility for users to personalize their image editing experience.

Here is an example of hiding the toolbar of the image editor using [toolbar](#).

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '350px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
            </div>
        </div>
        <script id="toolbarTemplate" type="text/x-template">
            <div class = 'e-toolbar'>
                <button id= 'dltbtn' class='e-btn e-primary'>Enable
FreeHandDraw</button>
            </div>
        </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide a toolbar Item

The [toolbar](#) property is utilized to control the visibility of toolbar items in the Image Editor. By default, the [toolbar](#) includes the default toolbar items. So, if you wish to hide the default toolbar items then you need to explicitly define the required items using [toolbar](#) property. This allows you to customize the toolbar by displaying only the specific items you require, tailoring the editing experience to your preferences.

Here is an example of hiding the cropping and selection toolbar items using [toolbar](#) property.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '350px',
    toolbar: ['Annotation', 'Finetune', 'Filter', 'Confirm', 'Reset',
'Save', 'ZoomIn', 'ZoomOut'],

```

```

created: () => {
    if (Browser.isDevice) {
        imageEditorObj.open('bee-eater.png');
    } else {
        imageEditorObj.open('bee-eater.png');
    }
}
});
imageEditorObj.appendTo('#imageeditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
            </div>
        </div>
        <script id="toolbarTemplate" type="text/x-template">
            <div class = 'e-toolbar'>

```

```

        <button id= 'dltbtn' class='e-btn e-primary'>Enable
FreeHandDraw</button>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable or disable a toolbar item

The [toolbar](#) property is employed to enable or disable toolbar items in the Image Editor. By default, the [toolbar](#) property includes the default toolbar items, and these items cannot be disabled. However, if you have defined custom toolbar items using the [toolbarItemModel](#), you can enable or disable them by configuring their respective properties within the [toolbar](#) property. This provides the flexibility to control the availability and functionality of custom toolbar items based on your specific requirements.

Here is an example of disabling the custom toolbar item using [toolbar](#) property.

Enable or disable a contextual toolbar item

The [toolbarItems](#) property in the [toolbarEventArgs](#) is used to enable or disable contextual toolbar items in the Image Editor. To enable or disable the default toolbar items, you can accomplish this by setting the [Disabled](#) property to true in the [ImageEditorToolbarItemModel](#) within the [ToolbarItems](#) property. This allows you to selectively enable or disable specific default toolbar items based on your requirements, providing a customized toolbar experience in the Image Editor.

Toolbar created event

The [toolbarCreated](#) event is triggered after the toolbar is created in the Image Editor. This event can be useful when you need to perform any actions or make modifications to the toolbar once it is fully initialized and ready for interaction. By subscribing to the [toolbarCreated](#) event, you can access the toolbar object and perform tasks such as adding event handlers, customizing the appearance, or configuring additional functionality.

Toolbar item clicked event

The [toolbarItemClicked](#) event is triggered when a toolbar item is clicked in the Image Editor. This event is particularly useful when you have added custom options to both the main toolbar and contextual toolbar, as it allows you to capture the user's interaction with those custom options. By subscribing to the [toolbarItemClicked](#) event, you can execute specific actions or handle logic based on the toolbar item that was clicked.

Here is an example of toolbar item clicking event using [toolbarItemClicked](#) property.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '350px',

```

```

toolbar: [{text: 'Custom'}],
created: () => {
    if (Browser.isDevice) {
        imageEditorObj.open('bee-eater.png');
    } else {
        imageEditorObj.open('bee-eater.png');
    }
},
toolbarItemClicked: (args: ClickEventArgs) => {
    if (args.item.text === 'Custom') {
        imageEditorObj.rotate(90);
    }
}
});
imageEditorObj.appendTo('#imageeditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">

```

```

        </div>
    </div>
</div>
<script id="toolbarTemplate" type="text/x-template">
    <div class = 'e-toolbar'>
        <button id= 'dltbtn' class='e-btn e-primary'>Enable
FreeHandDraw</button>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Toolbar template

The [toolbarTemplate](#) property in the Image Editor provides the capability to fully customize the toolbar by supplying a custom template. This feature is valuable when you want to create a distinct and personalized image editing experience that goes beyond the default toolbar or the customizable toolbar options offered by the Image Editor. By defining a custom template for the toolbar, you have complete control over its layout, appearance, and functionality. This empowers you to design a unique and tailored toolbar that aligns perfectly with your specific requirements and desired user experience.

Here is an example of using [toolbarTemplate](#) to render only the button to toggle the freehand draw option.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '350px',
    toolbarTemplate: '#toolbarTemplate',
    created: () => {
        if (Browser.isDevice) {
            imageEditorObj.open('bee-eater.png');
        } else {
            imageEditorObj.open('bee-eater.png');
        }
    }
});
imageEditorObj.appendTo('#imageeditor');
let button: Button = new Button({content: 'Enable
FreeHandDraw'}).appendTo('#dltbtn');
document.getElementById('dltbtn').onclick = (): void => {
    imageEditorObj.freeHandDraw(true);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
        </div>
      </div>
    </div>
    <script id="toolbarTemplate" type="text/x-template">
      <div class = 'e-toolbar'>
        <button id= 'dltbtn' class='e-btn e-primary'>Enable
FreeHandDraw</button>
      </div>
    </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


Customize Contextual Toolbar

The [toolbarUpdating](#) event is triggered when inserting or selecting annotations, which opens the contextual toolbar in the Image Editor. Within this event, the [toolbarItems](#) property in the [ToolbarEventArgs](#) is utilized to add or remove contextual toolbar items.

In the following example, the contextual toolbar for rectangle will be rendered with only stroke color by excluding fill color and stroke width using toolbarUpdating event.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '350px',
  toolbarUpdating: (args: ToolbarEventArgs) => {
    if (args.toolbarType === 'shapes') {
      args.toolbarItems = ['strokeColor'];
    }
  },
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
            </div>
        </div>
        <script id="toolbarTemplate" type="text/x-template">
            <div class = 'e-toolbar'>
                <button id= 'dltbtn' class='e-btn e-primary'>Enable
FreeHandDraw</button>
            </div>
        </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add an additional contextual Toolbar item to text shape

The contextual toolbar that appears when inserting annotations in the Image Editor is customizable using the [toolbarUpdating](#) event. This event is triggered when the contextual toolbar is rendered, allowing you to modify its contents. To add additional toolbar items to the contextual toolbar, you can access the [toolbarItems](#) property of the object within the event handler. By adding or removing items from the [toolbarItems](#) property based on the Item property, you can customize the options available in the contextual toolbar according to your needs. This gives you the ability to extend the functionality of the contextual toolbar and provide additional tools and options for working with inserted annotations.

Here is an example of adding the custom toolbar item to the contextual toolbar.

INDEX.TS

```

import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    created: () => {

```

```

    if (Browser.isDevice) {
        imageEditorObj.open('bee-eater.png');
    } else {
        imageEditorObj.open('bee-eater.png');
    }
},
toolbarUpdating: (args: ToolbarEventArgs) => {
    if (args.toolbarType === 'text') {
        args.toolbarItems.push({text: 'custom'})
    }
},
});
imageEditorObj.appendTo('#imageeditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
        <div>

```

```

        </div>
    </div>
    <script id="toolbarTemplate" type="text/x-template">
        <div class = 'e-toolbar'>
            <button id= 'dltbtn' class='e-btn e-primary'>Enable
FreeHandDraw</button>
        </div>
    </script>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Localization in the EJ2 JavaScript Image Editor control

The **Localization** library allows you to localize the default text content of the Image Editor. The Image Editor has static text that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the **locale** value and translation object.

The following list of properties and its values are used in the Image Editor.

Locale key words	Text
-----	-----
Browse	Browse
Crop	Crop
ZoomIn	Zoom In
ZoomOut	Zoom Out
Transform	Transform
Annotation	Annotation
Text	Add Text
Pen	Pen
Reset	Reset
Save	Save
Select	Select
RotateLeft	Rotate Left
RotateRight	Rotate Right
HorizontalFlip	Horizontal Flip
VerticalFlip	Vertical Flip
OK	OK

Cancel	Cancel
FillColor	Fill Color
StrokeColor	Stroke Color
StrokeWidth	StrokeWidth
FontFamily	Font Family
FontStyle	Font Style
FontSize	Font Size
FontColor	Font Color
Pan	Pan
Move	Move
Custom	Custom
Square	Square
Circle	Circle
Rectangle	Rectangle
Line	Line
Default	Default
Bold	Bold
Italic	Italic
BoldItalic	Bold Italic
XSmall	X-Small
Small	Small
Medium	Medium
Large	Large
XLarge	X-Large
ABC	ABC

INDEX.TS

```

import { L10n } from '@syncfusion/ej2-base';
import { ImageEditor } from '@syncfusion/ej2-image-editor';
L10n.load({
  'de-DE': {
    'image-editor': {
      'Browse': 'Durchsuche',
      'Crop': 'Ernte',
      'ZoomIn': 'Hineinzoomen',
      'ZoomOut': 'Rauszoomen',
      'Transform': 'Verwandeln',
      'Annotation': 'Anmerkung',
      'Text': 'Text hinzufügen',
    }
  }
});

```

```

        'Pen': 'Stift',
        'Reset': 'Zurücksetzen',
        'Save': 'Speichern',
        'Select': 'Auswählen',
        'RotateLeft': 'Nach links drehen',
        'RotateRight': 'Drehe nach rechts',
        'HorizontalFlip': 'Horizontaler Flip',
        'VerticalFlip': 'Vertikaler Flip',
        'OK': 'OK',
        'Cancel': 'Absagen',
        'FillColor': 'Füllfarbe',
        'StrokeColor': 'Strichfarbe',
        'StrokeWidth': 'Strichbreite',
        'FontFamily': 'Schriftfamilie',
        'FontStyle': 'Schriftstil',
        'FontSize': 'Schriftgröße',
        'FontColor': 'Schriftfarbe',
        'Pan': 'Pfanne',
        'Move': 'Bewegen',
        'Custom': 'Brauch',
        'Square': 'Quadrat',
        'Circle': 'Kreis',
        'Rectangle': 'Rechteck',
        'Line': 'Linie',
        'Default': 'Standard',
        'Bold': 'Fett gedruckt',
        'Italic': 'Kursiv',
        'BoldItalic': 'Fett Kursiv',
    }
}
});
//Image Editor items definition
new ImageEditor({
    locale: 'de-DE',
    width: '550px',
    height: '350px',
    }, '#imageeditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
            </div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filters in the EJ2 JavaScript Image Editor control

Filters are pre-defined effects that can be applied to an image to alter its appearance or mood. Image filters can be used to add visual interest or to enhance certain features of the image. Some common types of image filters include cold, warm, chrome, sepia, and invert. This can be done by either using the toolbar or the [applyImageFilter](#) method which takes a single parameter: the filter applied to an image.

Apply filter effect

The [applyImageFilter](#) method is utilized to apply filters to an image. By passing the desired filter type as the first parameter of the method, specified as [ImageFilterOption](#) the method applies the corresponding filter to the image. This allows for easy and convenient application of various filters to enhance or modify the image based on the chosen filter type.

- filterOption - Specifies the filter options to the image.

In the following example, you can using the applyImageFilter method in the button click event.

INDEX.TS

```

import { ImageEditor, ImageFilterOption } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    toolbar: [],
    created: () => {
        if (Browser.isDevice) {
            imageEditorObj.open('bee-eater.png');
        } else {
            imageEditorObj.open('bee-eater.png');
        }
    }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
document.getElementById('chromeClick').onclick = (): void => {
    imageEditorObj.applyImageFilter(ImageFilterOption.Chrome);
}
document.getElementById('coldClick').onclick = (): void => {
    imageEditorObj.applyImageFilter(ImageFilterOption.Cold);
}

document.getElementById('warmClick').onclick = (): void => {
    imageEditorObj.applyImageFilter(ImageFilterOption.Warm);
}
document.getElementById('grayscaleClick').onclick = (): void => {
    imageEditorObj.applyImageFilter(ImageFilterOption.Grayscale);
}
document.getElementById('sepiaClick').onclick = (): void => {
    imageEditorObj.applyImageFilter(ImageFilterOption.Sepia);
}
document.getElementById('invertClick').onclick = (): void => {
    imageEditorObj.applyImageFilter(ImageFilterOption.Invert);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
                <div>
                    <button id='chromeClick' class='e-btn e-primary'
>Chrome</button>
                    <button id='coldClick' class='e-btn e-primary'
>Cold</button>
                    <button id='warmClick' class='e-btn e-primary'
>Warm</button>
                    <button id='grayscaleClick' class='e-btn e-primary'
>Grayscale</button>
                    <button id='sepiaClick' class='e-btn e-primary'
>Sepia</button>
                    <button id='invertClick' class='e-btn e-primary'
>Invert</button>
                </div>
            </div>
        </div>
        <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
        </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Apply filter effect

The [imageFiltering](#) event is triggered when applying filtering on the image. This event is passed an object that contains information about the filtering event, such as the type of filtering.

The parameter available in the [ImageFilterEventArgs](#) event is,

`ImageFilterEventArgs.filter` - The type of filtering as [ImageFilterOption](#) to be applied in the image editor.

ImageFilterEventArgs.cancel – Specifies to cancel the filtering action.

Finetuning in the EJ2 JavaScript Image Editor control

Fine-tuning involves making precise adjustments to the settings of an image filter in order to achieve a specific desired effect. It provides control over the intensity and specific aspects of the filter's impact on the image. For example, fine-tuning allows you to modify parameters like brightness, saturation, or other relevant properties to fine-tune the level or quality of the filter's effect. This level of control enables you to achieve the exact look or outcome you want for your image.

Adjust the brightness, contrast, or sharpness

The [finetuneImage](#) method is designed to facilitate fine-tuning operations on an image. It accepts two parameters: the first parameter is [ImageFinetuneOption](#) which determines the type of fine-tuning to be applied (brightness, contrast, or sharpness), and the second parameter represents the fine-tuning value, indicating the degree or intensity of the adjustment. This method allows for convenient adjustment of brightness, contrast, or sharpness by specifying the desired type and corresponding value.

The [finetuneImage](#) method is used to perform brightness, contrast, or sharpness fine-tuning by specifying this type as a first parameter and specifying the fine-tuning value as the second parameter of the method.

Here is an example of brightness, contrast, and sharpness fine-tuning using the [finetuneImage](#) method.

- finetuneOption - Specifies the finetune options to be performed in the image.
- value - Specifies the value for finetuning the image.

In the following example, you can using the finetuneImage method in the button click event.

INDEX.TS

```
import { ImageEditor, ImageFinetuneOption } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
document.getElementById('brightnessClick').onclick = (): void => {
  imageEditorObj.finetuneImage(ImageFinetuneOption.Brightness,10);
}
document.getElementById('contrastClick').onclick = (): void => {
  imageEditorObj.finetuneImage(ImageFinetuneOption.Contrast,20);
}
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
        </div>
      <div>
        <button id='brightnessClick' class='e-btn e-primary'
>Brightness</button>
        <button id='contrastClick' class='e-btn e-primary'
>Contrast</button>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adjust the hue, exposure, blur, or opacity

The [finetuneImage](#) method is designed to facilitate fine-tuning operations on an image. It accepts two parameters: the first parameter is [ImageFinetuneOption](#) which determines the type of fine-tuning to be applied (hue, exposure, or blur), and the second parameter represents the fine-tuning value, indicating the degree or intensity of the adjustment. This method allows for convenient adjustment of hue, exposure, or blur by specifying the desired type and corresponding value.

Here is an example of hue, exposure, and blur fine-tuning using the [finetuneImage](#) method.

- finetuneOption - Specifies the finetune options to be performed in the image.
- value - Specifies the value for finetuning the image.

In the following example, you can using the finetuneImage method in the button click event.

INDEX.TS

```
import { ImageEditor, ImageFilterOption } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
document.getElementById('hueClick').onclick = (): void => {
  imageEditorObj.finetuneImage(ImageFinetuneOption.Hue, 20);
}
document.getElementById('exposureClick').onclick = (): void => {
  imageEditorObj.finetuneImage(ImageFinetuneOption.Exposure, 20);
}
document.getElementById('blurClick').onclick = (): void => {
  imageEditorObj.finetuneImage(ImageFinetuneOption.Blur, 20);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
                <div>
                    <button id='hueClick' class='e-btn e-primary' >Hue</button>
                    <button id='exposureClick' class='e-btn e-primary'
>Exposure</button>
                    <button id='blurClick' class='e-btn e-primary'
>Blur</button>
                </div>
            </div>
        </div>
        <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
        </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Finetune value changing event

The [finetuneValueChanging](#) event is triggered when performing fine-tuning on the image. This event is passed an object that contains information about the finetuning event, such as the type of fine-tuning and the value of fine-tuning performed.

The parameter available in the [FinetuneEventArgs](#) event is,

[FinetuneEventArgs.finetune](#) - The type of fine-tuning as [ImageFinetuneOption](#) to be applied in the image editor.

[FinetuneEventArgs.value](#) - The fine-tuning value to be applied in the image editor.

[FinetuneEventArgs.cancel](#) – Specifies a boolean value to cancel the fine-tuning action.

Frame in the EJ2 JavaScript Image Editor control

The frame feature in an Image Editor provides users with the capability to add decorative borders or frames around their images. Frames are a visual design element that can enhance the overall appearance and appeal of an image.

Apply frame to the Image

The [drawFrame](#) method is a function designed to enable the application of various frame options to an image. This method simplifies the process of adding decorative frames, such as mat, bevel, line, hook, and inset, to an image by allowing users to specify their desired frame type.

Depending on the frame type selected, users may have additional customization options, such as adjusting the frame's thickness, color, texture, or other attributes. This allows for fine-tuning the appearance of the frame to match the image's theme or the user's preferences

The [drawFrame](#) method in the Image Editor control takes nine parameters to define the properties of the frame to the image:

- [frameType](#) - Specified the image data or url of the image to be inserted.
- [Color](#) - Specifies the color for the frame.
- [gradientColor](#) - Specifies the gradient color for the frame.
- [size](#) - Specifies the size of the frame.
- [inset](#) - Specifies the inset value for line, hook, and inset type frames.
- [offset](#) - Specifies the offset value for line and inset type frames.
- [borderRadius](#) - Specifies the border radius for line type frame.
- [frameLineStyle](#) - Specifies the frame line style for line type frame.
- [lineCount](#) - Specifies the line count for the line type frame.

In the following example, you can use the [drawFrame](#) method in the button click event.

INDEX.TS

```
import { ImageEditor, FrameLineStyle, FrameType } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  toolbar: [],
  created: () => {
    imageEditorObj.open('bee-eater.png');
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
```

```

        document.getElementById('matClick').onclick = (): void => {
            imageEditorObj.drawFrame(FrameType.Mat, 'red', 'blue', 20, 20, 20,
20, FrameLineStyle.Solid, 1);
        }
        document.getElementById('bevelClick').onclick = (): void => {
            imageEditorObj.drawFrame(FrameType.Bevel, 'red', 'blue', 20, 20, 20,
20, FrameLineStyle.Solid, 1);
        }
        document.getElementById('lineClick').onclick = (): void => {
            imageEditorObj.drawFrame(FrameType.Line, 'red', 'blue', 20, 20, 20,
20, FrameLineStyle.Solid, 1);
        }
        document.getElementById('insetClick').onclick = (): void => {
            imageEditorObj.drawFrame(FrameType.Inset, 'red', 'blue', 20, 20, 20,
20, FrameLineStyle.Solid, 1);
        }
        document.getElementById('hookClick').onclick = (): void => {
            imageEditorObj.drawFrame(FrameType.Hook, 'red', 'blue', 20, 20, 20,
20, FrameLineStyle.Solid, 1);
        }
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

  <div id="container">
    <div class="control-section">
      <div id="imageeditor" class="row">
        </div>
      <div>
        <button id='matClick' class='e-btn e-primary' >Mat</button>
        <button id='bevelClick' class='e-btn e-primary'
>Bevel</button>
        <button id='lineClick' class='e-btn e-primary'
>Line</button>
        <button id='insetClick' class='e-btn e-primary'
>Inset</button>
        <button id='hookClick' class='e-btn e-primary'
>Hook</button>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Frame changing event

The [frameChanging](#) event is triggered when applying frame on the image. This event provides information encapsulated within an object, which includes details about the frame applied in an image. This information encompasses:

Frame Type: This indicates the specific type of frame being applied, whether it's a mat, bevel, line, or hook.

Customization Values: These values contain information about any adjustments or modifications made to the frame. For instance, if the frame can be customized with attributes like color, size, or style, these details are conveyed within the event object.

The parameter available in the [FrameChangeEventArgs](#) is

- [FrameChangeEventArgs.previousFrameSetting](#) - The frame settings including size, color, inset, offset, gradient color which is applied before changing the frame.
- [FrameChangeEventArgs.currentFrameSetting](#) - The frame settings including size, color, inset, offset, gradient color which is going to apply after changing the frame.
- [FrameChangeEventArgs.cancel](#) - Specifies a boolean value to cancel the frame changing action.

Resize

The resize feature in an Image Editor is a valuable tool that empowers users to modify the size or dimensions of an image to meet their specific requirements. Whether it's for printing, web display, or any other purpose, this feature allows users to tailor images to their desired specifications.

Apply resize to the image

The Image Editor control includes a [resize](#) method, which allows you to adjust the size of an image. This method takes three parameters that define how the resizing should be carried out:

- **width:** Specifies the resizing width of the image.
- **height:** Specifies the resizing height of the image.
- **isAspectRatio:** Specifies a boolean value indicating whether the image should maintain its original aspect ratio during resizing. When set to true, the image will be resized while preserving its aspect ratio

Here is an example of resizing the image using the [resize](#) method.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '330px',
    toolbar: [],
    created: () => {
        imageEditorObj.open('bee-eater.png');
    }
});
imageEditorObj.appendTo('#imageeditor');
document.getElementById('aspectClick').onclick = (): void => {
    imageEditorObj.resize(300, 400, true);
}
document.getElementById('nonaspectClick').onclick = (): void => {
    imageEditorObj.resize(400, 100, false);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                <div>
                    <button id='aspectClick' class='e-btn e-primary' >Aspect
Ratio</button>
                    <button id='nonaspectClick' class='e-btn e-primary' >Non
Aspect Ratio</button>
                </div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Resizing event

The [resizing](#) event is triggered when resizing the image. This event provides information encapsulated within an object, which includes details about the previous and current height and width of an image.

The parameter available in [ResizeEventArgs](#) is,

- [ResizeEventArgs.previousWidth](#) - The width of the image before resizing is performed.
- [ResizeEventArgs.previousHeight](#) - The height of the image before resizing is performed.
- [ResizeEventArgs.width](#) - The width of the image after resizing is performed.
- [ResizeEventArgs.height](#) - The width of the image after resizing is performed.
- [ResizeEventArgs.isAspectRatio](#) - The type of resizing performed such as aspect ratio or non-aspect ratio.
- [ResizeEventArgs.cancel](#) - Specifies a boolean value to cancel the resizing action.

Quick access toolbar in the EJ2 JavaScript Image Editor control

The quick access toolbars in the Image Editor play a vital role in facilitating interactions with annotations like Rectangle, Ellipse, Line, Arrow, and Path. These toolbars offer a diverse array of tools and options that can be tailored to match the specific requirements and preferences associated with each annotation type. The toolbar is only displayed when an annotation is selected, ensuring a focused and contextual user experience. Users have the flexibility to add or remove items from the toolbar, allowing them to create a personalized set of tools. Additionally, users can also build a completely custom toolbar from the ground up, providing them with complete control over the available options and functionality.

Add a custom toolbar item

The quick access toolbar that appears when inserting annotations in the Image Editor is customizable using the [quickAccessToolbarOpen](#) event. This event is triggered when the quick access toolbar is opened, allowing you to modify its contents. To add additional toolbar items to the quick access toolbar, you can access the `toolbarItems` property of the `QuickAccessToolbarEventArgs` within the event handler. By adding or removing items from the `toolbarItems` property based on the item property, you can customize the options available in the quick access toolbar according to your needs. This gives you the ability to extend the functionality of the quick access toolbar and provide additional tools and options for working with inserted annotations.

Here is an example of adding the custom toolbar item to the quick access toolbar.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
    width: '550px',
    height: '350px',
    quickAccessToolbarOpen: (args: quickAccessToolbarOpen) => {
        args.toolbarItems = ['Clone'];
    },
    created: () => {
        if (Browser.isDevice) {
            imageEditorObj.open('bee-eater.png');
        } else {
            imageEditorObj.open('bee-eater.png');
        }
    }
});
imageEditorObj.appendTo('#imageeditor');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                <div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Undo and Redo in the EJ2 JavaScript Image Editor control

The undo and redo functionalities provide a way to reverse and repeat editing actions performed on an image. These features are essential for maintaining control and flexibility during the editing process.

In an image editor, the undo and redo history typically have a limited capacity, and the number of steps that can be stored is 16 steps, meaning that the editor keeps track of the most recent 16 actions performed on the image. Once the history reaches its maximum capacity, any new actions beyond the 16th step will result in the removal of the oldest action from the history.

Undo the action

The undo action in an image editor allows users to revert the most recent editing action or a series of actions back to their previous state. When the undo command is triggered, the image editor undoes the last applied modification, effectively restoring the image to its state before the action was performed.

The undo action is useful for correcting mistakes, removing unwanted changes, or exploring different editing options without permanently altering the image.

Redo the action

The Redo action in an image editor allows users to reapply previously undone actions or modifications to the image. When the redo command is triggered, the image editor reapplies the last action that was undone, bringing the image back to the state it was in after the action was initially applied. The redo is useful when users want to repeat an action that was previously undone or restore changes that were temporarily reversed.

In the following example, the [undo](#) and [redo](#) method is used in the button click event.

INDEX.TS

```
import { ImageEditor } from '@syncfusion/ej2-image-editor';
import { Button } from '@syncfusion/ej2-buttons';
import { Browser } from '@syncfusion/ej2-base';
//Image Editor items definition
let imageEditorObj: ImageEditor = new ImageEditor({
  width: '550px',
  height: '330px',
  created: () => {
    if (Browser.isDevice) {
      imageEditorObj.open('bee-eater.png');
    } else {
      imageEditorObj.open('bee-eater.png');
    }
  }
});
imageEditorObj.appendTo('#imageeditor');
//Button click
let button: Button = new Button({cssClass: `e-primary`,
content: 'Click', '#btnClick'});
document.getElementById('undoClick').onclick = (): void => {
  imageEditorObj.undo();
}
document.getElementById('redoClick').onclick = (): void => {
  imageEditorObj.redo();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-image-
editor/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div id="imageeditor" class="row">
                </div>
            <div>
                <button id='undoClick' class='e-btn e-primary'
>Undo</button>
                <button id='redoClick' class='e-btn e-primary'
>Redo</button>
            </div>
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in EJ2 JavaScript Image Editor control

The Image Editor component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Image Editor component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | `` |

| Screen Reader Support | `` |

| Right-To-Left Support | `` |

| Color Contrast | `` |

| Mobile Device Support | `` |

| Keyboard Navigation Support | `` |

| [Accessibility Checker](#) Validation | `` |

| [Axe-core](#) Accessibility Validation | `` |

`<style>`

`.post .post-content img {`

`display: inline-block;`

`margin: 0.5em 0;`

`}`

`</style>`

`<div>` - All features of the component meet the requirement.`</div>`

`<div>` - Some features of the component do not meet the requirement.`</div>`

`<div>` - The component does not meet the requirement.`</div>`

Keyboard interaction

The Image Editor component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Image Editor component.

| **Press** | **To do this** |

| --- | --- |

| **Ctrl + Z** | **Undo the last user action.** |

| **Ctrl + Y** | **Redo the last user action.** |

| **Ctrl + S** | **To save the Image.** |

| Ctrl + O | To open the Image. |

| Delete | To delete the shape once the shape got selected through mouse click . |

Ensuring accessibility

The Image Editor component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Image Editor component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Image Editor component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

InPlaceEditor

Controls in EJ2 JavaScript In place editor control

In-place Editor renders various controls based on the [type](#) property and it have built-in and injectable controls. To use injectable controls, inject the required modules into **In-place Editor**. By default, the [type](#) property set to **Text** and render the **TextBox**.

The following table explains Injectable components module name and built-in components and their types.

Injectable Components Built in Components	
-----	-----
AutoComplete (AutoComplete)	TextBox (Text)
ComboBox (ComboBox)	DatePicker (Date)
MultiSelect (MultiSelect)	DateTimePicker (DateTime)
TimePicker (Time)	DropDownList (DropDownList)
DateRangePicker (DateRange)	MaskedTextBox (Mask)
Slider (Slider)	NumericTextBox (Numeric)
Rte (RTE)	
ColorPicker (Color)	

In the following sample, built-in and injectable based In-place Editor controls are rendered.

INDEX.TS

```
import { InPlaceEditor, AutoComplete, ColorPicker, ComboBox } from
 '@syncfusion/ej2-inplace-editor';
import { DateRangePicker, MultiSelect, Rte, Slider, TimePicker } from
 '@syncfusion/ej2-inplace-editor';
InPlaceEditor.Inject(AutoComplete, ColorPicker, ComboBox, DateRangePicker,
 MultiSelect, Rte, Slider, TimePicker);
let frameWorkList: string[] = ['Android', 'JavaScript', 'jQuery',
 'TypeScript', 'Angular', 'React', 'Vue', 'Ionic'];
```



```
let dateObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  type: 'Date',
  model: {
    placeholder: 'Select date'
  },
  value: new Date('11/23/2018')
});
dateObj.appendTo('#date');
let dateTimeObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  type: 'DateTime',
  model: {
    placeholder: 'Select date'
  },
  value: new Date('11/23/2018 12:30 PM')
});
dateTimeObj.appendTo('#dateTime');
let dropDownObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  type: 'DropDownList',
  value: 'Android',
  model: {
    dataSource: frameWorkList,
    placeholder: 'Select frameworks'
  }
});
dropDownObj.appendTo('#dropDowns');
let maskObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  type: 'Mask',
  value: '123-345-678',
  model: {
    mask: '000-000-000'
  }
});
maskObj.appendTo('#masked');
let numericObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  type: 'Numeric',
  value: 10,
  model: {
    placeholder: 'Enter number'
  }
});
numericObj.appendTo('#numeric');
let textObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  type: 'Text',
  value: 'Andrew',
  model: {
    placeholder: 'Enter some text'
  }
});
textObj.appendTo('#textbox');
let atcObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
```

```
        type: 'AutoComplete',
        value: 'Android',
        model: {
            dataSource: frameWorkList,
            placeholder: 'Select frameworks'
        }
    });
    atcObj.appendTo('#autoComplete');
    let colorObj: InPlaceEditor = new InPlaceEditor({
        mode: 'Inline',
        type: 'Color',
        value: '#81aefd'
    });
    colorObj.appendTo('#color');
    let comboBoxObj: InPlaceEditor = new InPlaceEditor({
        mode: 'Inline',
        type: 'ComboBox',
        value: 'Android',
        model: {
            dataSource: frameWorkList,
            placeholder: 'Select frameworks'
        }
    });
    comboBoxObj.appendTo('#comboBox');
    let dateRangeObj: InPlaceEditor = new InPlaceEditor({
        mode: 'Inline',
        type: 'DateRange',
        value: [new Date('11/12/2018'), new Date('11/15/2018')],
        model: {
            placeholder: 'Select date'
        }
    });
    dateRangeObj.appendTo('#dateRange');
    let multiSelectObj: InPlaceEditor = new InPlaceEditor({
        mode: 'Inline',
        type: 'MultiSelect',
        value: ['Android'],
        model: {
            dataSource: frameWorkList,
            placeholder: 'Select frameworks'
        }
    });
    multiSelectObj.appendTo('#multiSelect');
    let rteObj: InPlaceEditor = new InPlaceEditor({
        mode: 'Inline',
        type: 'RTE',
        value: '<p>Enter your content here</p>',
        model: {
            placeholder: 'Enter your content here'
        }
    });
    rteObj.appendTo('#rte');
    let sliderObj: InPlaceEditor = new InPlaceEditor({
        mode: 'Inline',
        type: 'Slider',
        value: 20
    });
```

```

sliderObj.appendTo('#slider');
let timeObj: InPlaceEditor = new InPlaceEditor({
    mode: 'Inline',
    type: 'Time',
    model: {
        placeholder: 'Select date'
    },
    value: new Date('11/23/2018')
});
timeObj.appendTo('#time');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript In-place Editor Control">
    <meta name="author" content="Syncfusion">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <h3> Built-in Controls </h3>
        <table class="table-section">
            <tbody><tr>
                <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> DatePicker </td>
                <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
                    <div id="date"></div>
                </td>
            </tr>
            <tr>
                <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> DateTimePicker </td>
                <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
                    <div id="dateTime"></div>
                </td>
            </tr>
            <tr>
                <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> DropDownList </td>
                <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
                    <div id="dropDowns"></div>
                </td>
            </tr>
        </tbody>
        </table>
    </div>

```

```

        </td>
    </tr>
    <tr>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> MaskedTextBox </td>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
            <div id="masked"></div>
        </td>
    </tr>
    <tr>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> NumericTextBox </td>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
            <div id="numeric"></div>
        </td></tr>
    <tr>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> TextBox </td>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
            <div id="textbox"></div>
        </td>
    </tr>
</tbody></table>
<h3> Injectable Controls </h3>
<table class="table-section">
    <tbody><tr>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> AutoComplete </td>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
            <div id="autoComplete"></div>
        </td>
    </tr>
    <tr>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> ColorPicker </td>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
            <div id="color"></div>
        </td>
    </tr>
    <tr>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> ComboBox </td>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
            <div id="comboBox"></div>
        </td>
    </tr>
    <tr>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> DateRangePicker </td>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
            <div id="dateRange"></div>
        </td>
    </tr>
    <tr>
        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> MultiSelect </td>

```

```

        <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
            <div id="multiSelect"></div>
        </td></tr>
        <tr>
            <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> RTE </td>
            <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
                <div id="rte"></div>
            </td>
        </tr>
        <tr>
            <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> Slider </td>
            <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
                <div id="slider"></div>
            </td>
        </tr>
        <tr>
            <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> TimePicker </td>
            <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
                <div id="time"></div>
            </td>
        </tr>
    </tbody></table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Model configuration

Control properties and events can be customized using the In-place Editor [model](#) property.

In the following code, the [type](#) defined as the `Date` and `DatePicker` properties are configured through [model](#) property to customize the [DatePicker](#) control at In-place Editor.

```

`ts
model: {
showTodayButton: true,
placeholder: 'Select Date'
}
`

```

[src/app/app.ts]

```

`ts

```

```
import { InPlaceEditor } from '@syncfusion/ej2-inplace-editor';
let editObj: InPlaceEditor = new InPlaceEditor({
  type: 'Date',
  value: new Date('04/12/2018'),
  model: {
    showTodayButton: true,
    placeholder: 'Select Date'
  }
});
editObj.appendTo('#element');
```

See Also

- [HTML5 components](#)

Configuration in EJ2 JavaScript In place editor control

Rendering modes

This section explains the supported rendering modes of the In-place Editor. Possible Rendering modes are as follows.

- Popup
- Inline

By default, **Popup** mode will be rendered, when opening an editor.

- For **Popup** mode, editable container displays as like tooltip or popover above the element.
- For **Inline** mode, editable container displays as instead of the element. To render **Inline** mode while opening the editor, specify **mode** as **Inline**.

In the following sample, the In-place Editor renders with **Inline** mode. You can dynamically switch into another mode by changing the drop-down item value.

INDEX.TS

```
import { InPlaceEditor, RenderMode } from '@syncfusion/ej2-inplace-editor';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
let modeData: string[] = ['Inline', 'Popup'];
let modeObj: DropDownList = new DropDownList({
  dataSource: modeData,
  width: 'auto',
  change: onChange,
  value: 'Inline',
  placeholder: 'Select Mode'
});
```

```

modeObj.appendTo('#dropDown');
let editObj: InPlaceEditor = new InPlaceEditor({
    mode: 'Inline',
    value: 'Andrew',
    model: {
        placeholder: 'Enter some text'
    }
});
editObj.appendTo('#element');
function onChange(e: ChangeEventArgs): void {
    let mode: RenderMode = e.itemData.value as RenderMode;
    /* Switch into another mode */
    editObj.mode = mode;
    editObj.dataBind();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 In-place Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript In-place Editor Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <table class="table-section">
            <tbody><tr>
                <td> Mode: </td>
                <td>
                    <div id="dropDown"></div>
                </td>
            </tr>
            <tr>
                <td class="sample-td"> Enter your name: </td>
                <td class="sample-td">
                    <div id="element"></div>
                </td>
            </tr>
        </tbody></table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pop-up customization

In-place Editor popup mode can be customized by using the [title](#) and [model](#) properties in [popupSettings](#) API.

Popup mode rendered by using the Essential JS 2 Tooltip control, so you can use tooltip properties and events to customize the behavior of popup via the [model](#) property of [popupSettings](#) API.

For more details, refer the tooltip documentation [section](#).

In the following sample, popup [title](#) and [position](#) customized using the [popupSettings](#) property. All possible tooltip position data configured in the drop-down, if we change drop down item, selected value bound to [model](#) property and applied it to [Tooltip](#) control. [Tooltip](#) have following position options.

- TopLeft
- TopCenter
- TopRight
- BottomLeft
- BottomCenter
- BottomRight
- LeftTop
- LeftCenter
- LeftBottom
- RightTop
- RightCenter
- RightBottom

INDEX.TS

```

import { InPlaceEditor } from '@syncfusion/ej2-inplace-editor';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
let positionData: string[] = ['TopLeft', 'TopCenter', 'TopRight',
'BottomLeft', 'BottomCenter', 'BottomRight', 'LeftTop', 'LeftCenter',
'LeftBottom', 'RightTop', 'RightCenter', 'RightBottom'];
let dropDownObj: DropDownList = new DropDownList({
    value: 'BottomCenter',
    dataSource: positionData,
    placeholder: 'Select a position',
    popupHeight: '150px',
    change: function(e: ChangeEventArgs): void {
        editObj.popupSettings.model.position = e.value;
        editObj.dataBind();
    }
});
dropDownObj.appendTo('#dropDown');
let editObj: InPlaceEditor = new InPlaceEditor({
    mode: 'Popup',
    value: 'Andrew',
    model: {

```



```

        placeholder: 'Enter some text'
    },
    popupSettings: {
        title: 'Enter name',
        model: {
            position: 'BottomCenter'
        }
    }
});
editObj.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 In-place Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript In-place Editor Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <table class="table-section">
            <tbody><tr>
                <td> Position: </td>
                <td>
                    <div id="dropDown"></div>
                </td>
            </tr>
            <tr>
                <td class="edit-heading sample-td"> Enter your name: </td>
                <td class="sample-td">
                    <div id="element"></div>
                </td>
            </tr>
        </tbody></table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Event actions for editing

The event action of the editor that enable in the edit mode based on the [editableOn](#) property, by default [Click](#) is assigned, the following options are also supported.

- [Click](#): The editor will be opened as single click actions.
- [DbClick](#): The editor will be opened as double-click actions and it is not applicable for edit icon.
- [EditIconClick](#): Disables the editing of event action of input and allows user to edit only through edit icon.

In-place Editor get focus by pressing the [tab](#) key from previous focusable DOM element and then by pressing [enter](#) key, the editor will be opened.

In the following sample, when switching drop-down item, the selected value assigned to the [editableOn](#) property. If you changed to [DbClick](#), the editor will open when making a double click on the input.

INDEX.TS

```
import { InPlaceEditor, EditableType } from '@syncfusion/ej2-inplace-editor';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
let editableOnData: string[] = ['Click', 'DbClick', 'EditIconClick'];
let dropDownObj: DropDownList = new DropDownList({
  dataSource: editableOnData,
  width: 'auto',
  value: 'Click',
  change: onChange,
  placeholder: 'Select edit type'
});
dropDownObj.appendTo('#dropDown');
let editObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  value: 'Andrew',
  model: {
    placeholder: 'Enter some text'
  }
});
editObj.appendTo('#element');
function onChange(e: ChangeEventArgs): void {
  let editType: EditableType = e.itemData.value as EditableType;
  editObj.editableOn = editType;
  editObj.dataBind();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 In-place Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript In-place Editor Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
```

```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <table class="table-section">
            <tbody><tr>
                <td> EditableOn: </td>
                <td>
                    <div id="dropDown"></div>
                </td>
            </tr>
            <tr>
                <td class="sample-td"> Enter your name: </td>
                <td class="sample-td">
                    <div id="element"></div>
                </td>
            </tr>
        </tbody></table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Action on focus out

Action to be performed when the user clicks outside the container, that means focusing out of editable content and it can be handled by the [actionOnBlur](#) property, by default **Submit** assigned. It also has the following options.

- **[Cancel](#)**: Cancels the editing and resets the old content.
- **[Submit](#)**: Submits the edited content to the server.
- **[Ignore](#)**: No action is performed with this type and allows to edit multiple editors.

In the following sample, when switching drop-down item, the selected value assigned to the [actionOnBlur](#) property.

INDEX.TS

```

import { InPlaceEditor, ActionBlur } from '@syncfusion/ej2-inplace-editor';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
let blurActionData: string[] = ['Submit', 'Cancel', 'Ignore'];
let dropDownObj: DropDownList = new DropDownList({
    dataSource: blurActionData,

```

```

        width: 'auto',
        value: 'Submit',
        change: onChange,
        placeholder: 'Select blur action'
    });
    dropDownObj.appendTo('#dropDown')
    let editObj: InPlaceEditor = new InPlaceEditor({
        mode: 'Inline',
        value: 'Andrew',
        model: {
            placeholder: 'Enter some text'
        }
    });
    editObj.appendTo('#element');
    function onChange(e: ChangeEventArgs): void {
        let editType: ActionBlur = e.itemData.value as ActionBlur;
        editObj.actionOnBlur = editType;
        editObj.dataBind();
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 In-place Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript In-place Editor Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <table class="table-section">
            <tbody><tr>
                <td> ActionOnBlur: </td>
                <td>
                    <div id="dropDown"></div>
                </td>
            </tr>
            <tr>
                <td class="sample-td"> Enter your name: </td>
                <td class="sample-td">
                    <div id="element"></div>
                </td>
            </tr>
        </tbody></table>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Display modes

By default, In-place Editor input element highlighted with a dotted underline. To remove dotted underline from input element, add `data-underline="false"` attribute at In-place Editor root element.

In the following sample, denotes to indicate intractable and normal display modes with different samples.

INDEX.TS

```
import { InPlaceEditor, ActionBlur } from '@syncfusion/ej2-inplace-editor';
let defaultObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  value: 'Andrew',
  model: {
    placeholder: 'Enter some text'
  }
});
defaultObj.appendTo('#default');
let inlineObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  value: 'Andrew',
  model: {
    placeholder: 'Enter some text'
  }
});
inlineObj.appendTo('#inline');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 In-place Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript In-place Editor Control">
  <meta name="author" content="Syncfusion">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <h4>Example of data-underline attribute</h4>
        <table class="table-section">
            <tbody><tr>
                <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> Intractable UI </td>
                <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
                    <div id="default"></div>
                </td>
            </tr>
            <tr>
                <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6 control-
title"> Normal UI </td>
                <td class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
                    <div id="inline" data-underline="false"></div>
                </td>
            </tr>
        </tbody></table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Disable the editor](#)
- [Animate the editor during popup mode](#)

Buttons in EJ2 JavaScript In place editor control

The In-place Editor had an action for save and cancel using buttons. The [saveButton](#) and [cancelButton](#) properties accept the [ButtonModel](#) objects for customizing the save and cancel button properties.

Buttons can be show or hide by sets a Boolean value to the [showButtons](#) property.

Without buttons value will be processed via the following ways.

- **[actionOnBlur](#)**: By clicking out side the editor control get focus out and do action based on this property value.
- **[submitOnEnter](#)**: Pressing **Enter** key it performs the submit action, if this property set to **true**.

In the following sample, the [content](#) and [cssClass](#) properties of [Button](#) value assigned to the [saveButton](#) and [cancelButton](#) properties to customize its appearance. Also check or uncheck a checkbox buttons render or removed from the editor.

To restrict either save or cancel button rendering into a DOM, simply pass empty object {} in the `saveButton` or `cancelButton` properties.

For more details about buttons, refer this documentation [section](#).

INDEX.TS

```
import { InPlaceEditor, ActionBlur } from '@syncfusion/ej2-inplace-editor';
import { CheckBox, ChangeEventArgs } from '@syncfusion/ej2-buttons';
let CheckBoxObj: CheckBox = new CheckBox({ label: 'Show', checked: true,
change: onChange });
CheckBoxObj.appendTo('#enableBtn');
let editObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  value: 'Andrew',
  model: {
    placeholder: 'Enter some text'
  },
  saveButton: {
    content: 'Ok',
    cssClass: 'e-outline'
  },
  cancelButton: {
    content: 'Cancel',
    cssClass: 'e-outline'
  }
});
editObj.appendTo('#element');
function onChange(e: ChangeEventArgs): void {
  editObj.showButtons = e.checked;
  editObj.dataBind();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 In-place Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript In-place Editor Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <table class="table-section">
      <tbody><tr>
```

```

        <td> ShowButtons: </td>
        <td>
            <input id="enableBtn" type="checkbox">
        </td>
    </tr>
    <tr>
        <td class="sample-td"> Enter your name: </td>
        <td class="sample-td">
            <div id="element"></div>
        </td>
    </tr>
</tbody></table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [In-place Editor buttons](#)

Server actions in EJ2 JavaScript In place editor control

By passing In-place Editor control value to the server, the [primaryKey](#) property value must require, otherwise action not performed for remote data.

If the [URL](#) property value is empty, data passing will handled at local and also the [actionSuccess](#) event will trigger with `null` as argument value.

The following arguments are passed to the server when submit actions perform.

Arguments	Explanations
value	For processing edited value, like DB value updating.
primaryKey	For value mapping to the server, like selecting DB.
name	For field mapping to the server, like DB column field mapping.

Find the following sample server codes for defining models and controller functions to configure processing data.

```

`C#
public class SubmitModel
{
    public string Name { get; set; }
    public string PrimaryKey { get; set; }
}

```



```

public string Value { get; set; }
}
`C#
public IEnumerable<SubmitModel> UpdateData([FromBody]SubmitModel value)
{
    // User can process data
    return value;
}
`

```

- Server actions successfully done, the [actionSuccess](#) event will be fired with returned server data.
- If the server is not responding, the [actionFailure](#) event will be fired with data, but value not updated in the Editor.

In the following sample, the `actionSuccess` event will trigger once the value submitted successfully into the server. In this sample, both `actionSuccess` and `actionFailure` were configured and resulted value will be converted to chips.

INDEX.TS

```

import { InPlaceEditor, ActionEventArgs, MultiSelect } from
 '@syncfusion/ej2-inplace-editor';
InPlaceEditor.Inject(MultiSelect);
let serviceUrl: string =
 'https://ej2services.syncfusion.com/development/web-
 services/api/Editor/UpdateData';
let frameWorkList: string[] = ['Android', 'JavaScript', 'jQuery',
 'TypeScript', 'Angular', 'React', 'Vue', 'Ionic'];
let editObj: InPlaceEditor = new InPlaceEditor({
    mode: 'Inline',
    type: 'MultiSelect',
    value: ['JavaScript', 'jQuery'],
    name: 'skill',
    url: serviceUrl,
    primaryKey: "FrameWork",
    adaptor: 'UrlAdaptor',
    model: {
        mode: 'Box',
        dataSource: frameWorkList,
        placeholder: 'Select skill'
    },
    actionSuccess: onActionSuccess,
    actionFailure: onActionFailure
});
editObj.appendTo('#element');
chipOnCreate(); // Initialize chips customization at initial load
function chipOnCreate(): void {

```

```

    editObj.element.querySelector('.e-editable-value').innerHTML =
    chipCreation(editObj.value);
}
function onActionSuccess(e: ActionEventArgs): void {
    e.value = chipCreation(e.value.split(','), true);
}
function onActionFailure(e: ActionEventArgs): void {
    e.value = chipCreation(e.value.split(','), false);
}
function chipCreation(data: string[], isSuccess: boolean): string {
    let value: string = '<div class="e-chip-list">';
    [].slice.call(data).forEach((val: string) => {
        value += '<div class="e-chip"> <span class="e-chip-text' +
        (!isSuccess ? 'e-highlight' : '') + '> ' + val + '</span></div>';
    });
    value += '</div>';
    return value;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 In-place Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript In-place Editor Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
    rel="stylesheet">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <span class="content-title"> Select frameworks: </span>
        <div id="element" data-underline="false"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Indicate the server actions in the editor](#)

Data binding in EJ2 JavaScript In place editor control

The Essential JS 2 controls load the data either from local data sources or remote data services using the `dataSource` property and it supports the data type of an array or `DataManager`. Also supports different kind of data services such as OData, OData V4, Web API, and data formats such as XML, JSON, JSONP with the help of `DataManager` adaptors.

Local

To bind local data to the Essential JS 2 controls, you can assign a JavaScript array of object or string to the `dataSource` property. The local data source can also be provided as an instance of the `DataManager`.

INDEX.TS

```
import { InPlaceEditor } from '@syncfusion/ej2-inplace-editor';
let gameList: { [key: string]: Object }[] = [
  { Id: '1', Name: 'Maria Anders' },
  { Id: '2', Name: 'Ana Trujillo' },
  { Id: '3', Name: 'Antonio Moreno' },
  { Id: '4', Name: 'Thomas Hardy' },
  { Id: '5', Name: 'Chiristina Berglund' },
  { Id: '6', Name: 'Hanna Moos' }
];
let editObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  type: 'DropDownList',
  value: 'Maria Anders',
  model: {
    dataSource: gameList,
    fields: { text: 'Name' },
    placeholder: 'Select a customer'
  }
});
editObj.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 In-place Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript In-place Editor Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

        <span class="content-title"> Select customer name: </span>
        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Remote

To bind remote data to the Essential JS 2 control, assign service data as an instance of **DataManager** to the **dataSource** property. To interact with remote data source, provide the endpoint URL.

OData V4

The OData V4 is an improved version of OData protocols, and the **DataManager** can also retrieve and consume OData V4 services. To fetch data from the server by using **DataManager** with the adaptor property configure as **ODataV4Adaptor**.

In the following sample, In-place Editor renders a **DropDownList** control and its **dataSource** property configured for fetching a data from the server by using **ODataV4Adaptor** with **DataManager**.

INDEX.TS

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { InPlaceEditor } from '@syncfusion/ej2-inplace-editor';
let editObj: InPlaceEditor = new InPlaceEditor({
    mode: 'Inline',
    type: 'DropDownList',
    value: 'Maria Anders',
    model: {
        dataSource: new DataManager({
            url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
            adaptor: new ODataV4Adaptor,
            crossDomain: true
        }),
        placeholder: "Select a customer",
        query: new Query().from('Customers').select(['ContactName',
        'CustomerID']).take(6),
        fields: { text: 'ContactName', value: 'CustomerID' }
    }
});
editObj.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 In-place Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript In-place Editor Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <span class="content-title"> Select customer name: </span>
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Web API

Data can fetch from the server by using [DataManager](#) with the adaptor property configure as [WebApiAdaptor](#).

In the following sample, In-place Editor render a **DropDownList** control and its **dataSource** property configured for fetching a data from the server by using **WebApiAdaptor** with **DataManager**.

INDEX.TS

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { InPlaceEditor } from '@syncfusion/ej2-inplace-editor';
let editObj: InPlaceEditor = new InPlaceEditor({
    mode: 'Inline',
    type: 'DropDownList',
    value: 'Maria Anders',
    model: {
        dataSource: new DataManager({
            url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
            adaptor: new ODataV4Adaptor,
        }),
        placeholder: "Select a customer",
        fields: { text: 'ContactName', value: 'CustomerID' }
    }
});
editObj.appendTo('#element');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 In-place Editor</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript In-place Editor Control">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <span class="content-title"> Select customer name: </span>
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Integration in EJ2 JavaScript In place editor control

The In-place Editor supports adding HTML5 input controls using the [template](#) property. The Template property can be given as either a [string](#) or a [query selector](#).

As a string

The HTML element tag can be given as a string for the template property. Here, the input is rendered as an HTML template.

```
`ts
```

```
template: "<div><input type='text' id='name'></input></div>"
```

```
,
```

As a selector

The template property also allows getting template content through query [selector](#). Here, the input wrapper element 'ID' attribute is specified in the template.

```
`ts
```

```
template: "#date"
```

```
,
```

Template mode, the [value](#) property not handled by the In-place Editor control. So, before sending a value to the server, you need to modify at [actionBegin](#) event, otherwise, an empty string will pass. In the

following template sample, before submitting a data to the server, event argument and [value](#) property content updated in the `actionBegin` event handler.

INDEX.TS

```
import { InPlaceEditor, ActionBeginEventArgs } from '@syncfusion/ej2-inplace-editor';
let editObj: InPlaceEditor = new InPlaceEditor({
    mode: 'Inline',
    template: '#date',
    value: '2018-05-23',
    actionBegin: function(e: ActionBeginEventArgs): void {
        let value: string =
        (<HTMLInputElement>editObj.element.querySelector('#date')).value;
        editObj.value = value;
        e.value = value;
    }
});
editObj.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 In-place Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript In-place Editor Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <span class="content-title"> Select date: </span>
        <div id="element"></div>
    </div>
    <div id="html-template" style="display: none">
        <input id="date" value="2018-05-23" type="date">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [Built-in Controls](#)

Validation in EJ2 JavaScript In place editor control

In-place Editor control supports validation and it can be achieved by adding rules to the [validationRules](#) property, its child property `key` must be same as [name](#) property, otherwise validation not performed. Submitting data to the server or calling the [validate](#) method validation executed.

Validation Rules

In-place Editor has following validation rules, which are used to perform validation.

Rules	Description	Example
----- ----- -----		
required	The input element must have any input values	a or 1 or -
email	The input element must have valid email format values	<inplace@syncfusion.com>
url	The input element must have valid url format values	http://syncfusion.com/
date	The input element must have valid date format values	12/25/2019
dateIso	The input element must have valid dateIso format values	2019-12-25
number	The input element must have valid number format values	1.0 or 1
maxLength	Input value must have less than or equal to maxLength character length	if maxLength : 5, [check] is valid and [checking] is invalid
minLength	Input value must have less than or equal to minLength character length	if minLength : 5, [testing] is valid and [test] is invalid
rangeLength	Input value must have value between rangeLength character length	if rangeLength : [4,5], [test] is valid and [key] is invalid
range	Input value must have value between range number	if range : [4,5], [4] is valid and [6] is invalid
max	Input value must have less than or equal to max number	if max : 3, [3] is valid and [4] is invalid
min	Input value must have less than or equal to min number	if min : 4, [5] is valid and [2] is invalid

Accessibility in EJ2 JavaScript Inplace editor component

Style in EJ2 JavaScript In place editor control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the In-place Editor text

Use the following CSS to customize the default In-place Editor's text content properties like font-family, font-size, color and border bottom.


```
`css
/ To change color, font family and font size /
.e-inplaceeditor .e-editable-value-wrapper .e-editable-value {
border-bottom: 2px dotted green;
color: red;
font-size: 12px;
font-family: Segoe UI
}
`
```

[Customizing the In-place Editor action buttons](#)

Use the following CSS to customize the default In-place Editor's action buttons.

```
`css
/ To change icon color for save button /
.e-inplaceeditor .e-editable-action-buttons .e-btn-save.e-icon-btn .e-btn-icon.e-icons,
.e-inplaceeditor-tip .e-editable-action-buttons .e-btn-save.e-icon-btn .e-btn-icon.e-icons{
color: green;
}

/ To change icon color for cancel button /
.e-inplaceeditor .e-editable-action-buttons .e-btn-cancel.e-icon-btn .e-btn-icon.e-icons, .e-
inplaceeditor-tip .e-editable-action-buttons .e-btn-cancel.e-icon-btn .e-btn-icon.e-icons {
color: red;
}

/ To change background color for save button /
.e-inplaceeditor .e-editable-action-buttons .e-btn-save.e-icon-btn,
.e-inplaceeditor-tip .e-editable-action-buttons .e-btn-save.e-icon-btn {
background-color: antiquewhite;
}

/ To change background color for cancel button /
.e-inplaceeditor .e-editable-action-buttons .e-btn-cancel.e-icon-btn,
.e-inplaceeditor-tip .e-editable-action-buttons .e-btn-cancel.e-icon-btn {
background-color: antiquewhite;
}
`
```

Localization in EJ2 JavaScript In place editor control

Localization

Localization library allows you to localize the default text content of the In-place Editor to different cultures using the [locale](#) property. In-place Editor following keys will be localize based on culture.

| Locale key | en-US (default) |

|-----|-----|

| save | Close |

| cancel | Cancel |

| loadingText | Loading... |

| editIcon | Click to edit |

| editAreaClick | Click to edit |

| editAreaDoubleClick | Double click to edit |

To load translation object in an application use `load` function of `L10n` class. In the following sample, **French** culture is set to In-place Editor and change the tooltip text.

INDEX.TS

```
import { L10n } from '@syncfusion/ej2-base';
import { InPlaceEditor, EditableType } from '@syncfusion/ej2-inplace-editor';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
L10n.load({
    'fr-BE': {
        'inplace-editor': {
            'save': 'enregistrer',
            'cancel': 'Annuler',
            'loadingText': 'Chargement...',
            'editIcon': 'Cliquez pour éditer',
            'editAreaClick': 'Cliquez pour éditer',
            'editAreaDoubleClick': 'Double-cliquez pour éditer'
        }
    }
});
let editableOnData: string[] = ['Click', 'DbClick', 'EditIconClick'];
let dropDownObj: DropDownList = new DropDownList({
    dataSource: editableOnData,
    width: 'auto',
    value: 'Click',
    change: onChange,
    placeholder: 'Select edit type'
});
dropDownObj.appendTo('#dropDown');
let editObj: InPlaceEditor = new InPlaceEditor({
    mode: 'Inline',
    value: 'Andrew',
    locale: 'fr-BE',
    model: {
        placeholder: 'Enter some text'
    }
});
```

```
});
editObj.appendTo('#element');
function onChange(e: ChangeEventArgs): void {
    let editType: EditableType = e.itemData.value as EditableType;
    editObj.editableOn = editType;
    editObj.dataBind();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 In-place Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript In-place Editor Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <table class="table-section">
            <tbody><tr>
                <td> EditableOn: </td>
                <td>
                    <div id="dropDown"></div>
                </td>
            </tr>
            <tr>
                <td class="sample-td"> Enter your name: </td>
                <td class="sample-td">
                    <div id="element"></div>
                </td>
            </tr>
        </tbody></table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Right to left

Specifies the direction of the In-place Editor control using the enableRtl property. For writing systems that require it like Arabic, Hebrew, etc., the direction can be switched to right-to-left.

It will not change based on the locale property.

INDEX.TS

```
import { InPlaceEditor } from '@syncfusion/ej2-inplace-editor';
let editObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  value: 'Andrew',
  enableRtl: true
});
editObj.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 In-place Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript In-place Editor Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <span class="content-title"> Enter your name: </span>
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Format

Formatting is a way of representing the value in different format. You can format the following mentioned controls with its `format` property, when it passed through the In-place Editor [model](#) property.

- [DatePicker](#)
- [DateRangePicker](#)
- [DateTimePicker](#)
- [NumericTextBox](#)
- [Slider](#)
- [TimePicker](#)

INDEX.TS

```
import { InPlaceEditor } from '@syncfusion/ej2-inplace-editor';
let editObj: InPlaceEditor = new InPlaceEditor({
  type: 'Date',
  value: new Date('11/23/2018'),
  model: {
    placeholder: 'Select date',
    format: 'yyyy-MM-dd'
  }
});
editObj.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 In-place Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript In-place Editor Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <span class="content-title"> Select date: </span>
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

How To

Dynamic edit mode in EJ2 JavaScript In place editor control

At control initial load, if you want to open editor state without interacting In-place Editor input element, it can be achieved by configuring the [enableEditMode](#) property to [true](#).

In the following sample, editor opened at initial load and when toggling a checkbox, it will remove or open the editor.

INDEX.TS

```
import { InPlaceEditor, ActionBlur } from '@syncfusion/ej2-inplace-editor';
import { CheckBox, ChangeEventArgs } from '@syncfusion/ej2-buttons';
let CheckBoxObj: CheckBox = new CheckBox({ label: 'Enable', checked: true,
change: onChange });
CheckBoxObj.appendTo('#enable');
let editObj: InPlaceEditor = new InPlaceEditor({
    mode: 'Inline',
    value: 'Andrew',
    enableEditMode: true,
    actionOnBlur: 'Ignore'
    model: {
        placeholder: 'Enter some text'
    }
});
editObj.appendTo('#element');
function onChange(e: ChangeEventArgs): void {
    editObj.enableEditMode = e.checked;
    editObj.dataBind();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 In-place Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript In-place Editor Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <table class="table-section">
            <tbody><tr>
                <td> EnableEditMode: </td>
                <td>
                    <input id="enable" type="checkbox">
                </td>
            </tr>
        </tbody>
    </table>
    </div>
</body>
```

```

        </td>
      </tr>
      <tr>
        <td class="sample-td"> Enter your name: </td>
        <td class="sample-td">
          <div id="element"></div>
        </td>
      </tr>
    </tbody></table>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable edit mode in EJ2 JavaScript In place editor control

The edit mode of In-place Editor can be disabled by setting the [disabled](#) property value to **true**. In the following sample, when check or uncheck the checkbox, In-place Editor control will disable or enable the edit mode.

INDEX.TS

```

import { InPlaceEditor } from '@syncfusion/ej2-inplace-editor';
import { CheckBox, ChangeEventArgs } from '@syncfusion/ej2-buttons';
let CheckBoxObj: CheckBox = new CheckBox({ label: 'Disable', checked: false,
change: onChange });
CheckBoxObj.appendTo('#enable');
let editObj: InPlaceEditor = new InPlaceEditor({
  mode: 'Inline',
  value: 'Andrew',
  model: {
    placeholder: 'Enter some text'
  }
});
editObj.appendTo('#element');
function onChange(e: ChangeEventArgs): void {
  editObj.disabled = e.checked;
  editObj.dataBind();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 In-place Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript In-place Editor Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <table class="table-section">
            <tbody><tr>
                <td> Disabled: </td>
                <td>
                    <input id="enable" type="checkbox">
                </td>
            </tr>
            <tr>
                <td class="sample-td"> Enter your name: </td>
                <td class="sample-td">
                    <div id="element"></div>
                </td>
            </tr>
        </tbody></table>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom indication in EJ2 JavaScript In place editor control

You can add custom indication to unsaved input value by using the [actionSuccess](#) event, when data not submitted to the server.

In this sample, the `actionSuccess` event configured and the `URL` property not included. Then submit button clicked, the current editor value saved into input and data sending to server action prevented due to the `URL` property not configured.

But `actionSuccess` event will trigger the handler function with `null` argument values. In handler function data property `primaryKey` value checked, whether it empty or not. If it is empty custom class, added in the `e-value-wrapper` element to customize its styles.

To send input value to local, set the `URL` property as empty.

INDEX.TS

```

import { isNullOrUndefined as isNOU } from '@syncfusion/ej2-base';
import { InPlaceEditor, ActionEventArgs } from '@syncfusion/ej2-inplace-editor';
let editObj: InPlaceEditor = new InPlaceEditor({
    mode: 'Inline',

```



```

    value: 'Andrew',
    model: {
        placeholder: 'Enter some text'
    },
    actionSuccess: function(e: ActionEventArgs): void {
        let pk: string = e.data['PrimaryKey'];
        if (isNOU(pk) || pk === '') {
            document.querySelector('.e-editable-value').classList.add('e-send-error');
        }
    }
});
editObj.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 In-place Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript In-place Editor Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <span class="content-title"> Enter your name: </span>
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom animation in EJ2 JavaScript In place editor control

In popup mode, the In-place Editor rendered with the Essential JS 2 **Tooltip** control. You can use tooltip properties and events to customize the popup by configure properties into the [model](#) property inside the [popupSettings](#) API.

In the following sample, popup animation can be customized by passing animation effect using the `model` property and the dynamic animation effect changes configured from the Essential JS 2 `DropDownList` control `change` event.

INDEX.TS

```
import { InPlaceEditor } from '@syncfusion/ej2-inplace-editor';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
var openAnimateData = ['None', 'FadeIn', 'FadeZoomIn', 'ZoomIn'];
let openDropObj: DropDownList = new DropDownList({
    value: 'ZoomIn',
    dataSource: openAnimateData,
    placeholder: 'Select a animate type',
    popupHeight: '150px',
    change: function(e: ChangeEventArgs): void {
        editObj.popupSettings.model.animation.open.effect = e.value;
        editObj.dataBind();
    }
});
openDropObj.appendTo('#openDropDown');
let editObj: InPlaceEditor = new InPlaceEditor({
    mode: 'Popup',
    value: 'Andrew',
    model: {
        placeholder: 'Enter some text'
    },
    popupSettings: {
        model: {
            animation: {
                open: { effect: 'ZoomIn', duration: 1000, delay: 0 }
            }
        }
    }
});
editObj.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 In-place Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript In-place Editor Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```
<div id="container">
  <table class="table-section">
    <tbody><tr>
      <td> Open Animation: </td>
      <td>
        <div id="openDropDown"></div>
      </td>
    </tr>
    <tr>
      <td class="sample-td"> Enter your name: </td>
      <td class="sample-td">
        <div id="element"></div>
      </td>
    </tr>
  </tbody></table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Kanban

Getting started in EJ2 JavaScript Kanban control

This section briefly explains how to create the **Kanban** component and configure its available functionalities in a JavaScript application.

Dependencies

The following list of dependencies are required to use the Kanban component in your application:

`javascript

```
|-- @syncfusion/ej2-kanban
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-icons
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-layouts
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-popups
```

```
|-- @syncfusion/ej2-splitbuttons
`
```

Setup for local environment

Refer to the following steps to setup your local environment:

Step 1: Create a root folder `myapp` for your application.

Step 2: Create `myapp/resources` folder to store local scripts and styles files.

Step 3: Create `myapp/index.js` and `myapp/index.html` files for initializing Essential JS 2 Kanban control.

Adding Syncfusion resources

The Essential JS 2 Kanban control can be initialized by using one of the following ways.

- Using local scripts and styles.
- Using CDN links for scripts and styles.

Using local scripts and styles

You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

After installing the Essential JS 2 product build, you can copy the Kanban and its dependency scripts and style files into the `resources/scripts` and `resources/styles` folder respectively.

Refer to the following location from where the Kanban's script and styles can be referenced.

Syntax:

Script: `(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js`

Styles: `(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css`

Example:

Script: `C:/Program Files (x86)/Syncfusion/Essential Studio/17.4.39/Essential JS 2/ej2-kanban/dist/global/ej2-kanban.min.js`

Styles: `C:/Program Files (x86)/Syncfusion/Essential Studio/17.4.39/Essential JS 2/ej2-kanban/styles/material.css`

After copying the files, you can refer the Kanban's scripts and styles into the `index.html` file. The following HTML code example shows the dependency of Kanban.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<title>Essential JS 2 Kanban</title>
<!-- Essential JS 2 Kanban's dependent material theme -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/buttons/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/dropdowns/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/inputs/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/layouts/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/popups/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/navigations/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Kanban's material theme -->
<link href="resources/kanban/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Kanban's dependent scripts -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-buttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-dropdowns.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-inputs.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-splitbuttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-lists.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-navigations.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-layouts.min.js" type="text/javascript"></script>
<!-- Essential JS 2 Kanban's global script -->
<script src="resources/scripts/ej2-kanban.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
```

Using CDN links for scripts and styles

Using CDN link, you can directly refer the Kanban's script and styles into the `index.html` file.

Refer to the Kanban's CDN links given as follows.

Syntax:

Script: `http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Styles: `http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

Example:

Script: <http://cdn.syncfusion.com/ej2/ej2-kanban/dist/global/ej2-kanban.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-kanban/styles/material.css>

The following HTML code example shows the dependency of Kanban with `ej2-kanban.min.js`.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Kanban</title>
<!-- Essential JS 2 Kanban's dependent material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-layouts/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Kanban's material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-kanban/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Kanban's dependent script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>
```

```

<script src="http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-dropdown.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-inputs/dist/global/ej2-inputs.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-buttons/dist/global/ej2-buttons.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-layouts/dist/global/ej2-layouts.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-splitbuttons/dist/global/ej2-splitbuttons.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-lists/dist/global/ej2-lists.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-popups.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>
<!-- Essential JS 2 Kanban's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-kanban/dist/global/ej2-kanban.min.js"
type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element for kanban -->
<div id="Kanban"></div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Place the following Kanban code in the `index.js` file.

```

`javascript
var kanbanObj = new ej.kanban.Kanban({
columns: [
{ headerText: 'Backlog', keyField: 'Open' },
{ headerText: 'In Progress', keyField: 'InProgress' },
{ headerText: 'Testing', keyField: 'Testing' },

```

```
{ headerText: 'Done', keyField: 'Close' }
]
});
kanbanObj.appendTo('#Kanban');
`
```

Initialize the Kanban

Now, you can add the Kanban control to the application. For getting started, add a `div` element for Kanban control in `index.html`. Then refer the `index.js` file into the `index.html` file.

In this document context, we are going to use `ej2.min.js`, which includes all the Essential JS 2 components and their dependent scripts.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Kanban</title>
<!-- Essential JS 2 Kanban's dependent material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-layouts/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Kanban's material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-kanban/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 all script -->
<script src="http://cdn.syncfusion.com/ej2/dist/ej2.min.js" type="text/javascript"></script>
</head>
```



```

<body>
<!-- Add the HTML <div> element for kanban -->
<div id="Kanban"></div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>
,

```

Place the following Kanban code in the `index.js` file.

```

`javascript
var kanbanObj = new ej.kanban.Kanban({
columns: [
{ headerText: 'Backlog', keyField: 'Open' },
{ headerText: 'In Progress', keyField: 'InProgress' },
{ headerText: 'Testing', keyField: 'Testing' },
{ headerText: 'Done', keyField: 'Close' }
]
});
kanbanObj.appendTo('#Kanban');
,

```

INDEX.JS

```

var kanbanObj = new ej.kanban.Kanban({
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ]
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Columns Header</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban Columns Header">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Populating cards

To populate the empty Kanban with cards, define the local JSON data or remote data using the `dataSource` property. To define `dataSource`, the mandatory fields in JSON object should be relevant to `keyField`. In the following example, you can see the cards defined with default fields such as ID, Summary, and Status.

INDEX.JS

```

var kanbanObj = new ej.kanban.Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {

```

```
        contentField: 'Summary',
        headerField: 'Id',
    }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Cards</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban Cards">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Enable swimlane

Swimlane can be enabled by mapping the fields `swimlaneSettings.keyField` to appropriate column name in `dataSource`. This enables the grouping of the cards based on the mapped column values.

INDEX.JS

```
var kanbanObj = new ej.kanban.Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id',
  },
  swimlaneSettings: {
    keyField: 'Assignee'
  }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Kanban with Swimlane Rows</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban swimlane rows">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Columns in EJ2 JavaScript Kanban control

The **Kanban** columns represent the each stage of the process. The column definitions are used as the **dataSource** schema in the Kanban. The Kanban operations such as drag-and-drop, swimlane, and toggle columns are performed based on column definitions.

Single-key mapping

Kanban columns are categorized by mapping the **key** from the datasource using the **keyField** property. The corresponding **value** in the datasource is mapped inside the columns **keyField**. Based on this categorization, Kanban columns are split on this board.

The **keyField** property is mandatory to render the columns in the Kanban board.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Single Key Columns</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Kanban single key columns">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multi-key mapping

Kanban board allows to render a single column by mapping multiple keys using `keyField` property. In below sample, specified the multiple keys(Open, Validate) to a single column.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',

```

```

columns: [
    { headerText: 'Backlog', keyField: 'Open, Validate' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
],
cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
}
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Multiple Key Columns</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban columns with multiple keys">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```
    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Header text

You can provide the column header text of Kanban columns using the `headerText` property. If you have not specified any header text, it will render the header without any text.

Header template

You can customize the column header with any HTML or CSS element using the `template` property as shown in the following code.

You can get the following columns data when using header template.

- `keyField`
- `headerText`
- `minCount`
- `maxCount`
- `allowToggle`
- `isExpanded`
- `showItemCount`
- `count`

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open', template:
    '#headerTemplate' },
    { headerText: 'In Progress', keyField: 'InProgress', template:
    '#headerTemplate' },
    { headerText: 'Review', keyField: 'Review', template:
    '#headerTemplate' },
    { headerText: 'Done', keyField: 'Close', template: '#headerTemplate'
  }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Column Header Template</title>
```



```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Kanban column header template">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script id="headerTemplate" type="text/x-jsrender">
  <div class="header-template-wrap">
    <div class="header-icon e-icons ${keyField}"></div>
    <div class="header-text">${headerText}</div>
  </div>
</script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <style type="text/css">
    .e-kanban .header-template-wrap {
      display: inline-flex;
      font-size: 15px;
      font-weight: 400;
    }
    .e-kanban .header-template-wrap .header-icon {
      font-family: 'Kanban priority icons';
      margin-top: 3px;
      width: 10%;
    }
    .e-kanban .header-template-wrap .header-text {
      margin-left: 15px;
    }
    .e-kanban .card-header {
      padding-left: 12px;
    }
    .e-kanban .Open::before {
      content: '\e700';
      color: #0251cc;
      font-size: 16px;

```

```
}
.e-kanban .InProgress::before {
    content: '\e703';
    color: #ea9713;
    font-size: 16px;
}
.e-kanban .Review::before {
    content: '\e701';
    color: #8e4399;
    font-size: 16px;
}
.e-kanban .Close::before {
    content: '\e702';
    color: #63ba3c;
    font-size: 16px;
}
@font-face {
    font-family: 'Kanban priority icons';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAAwAgTlMvMjltSfUAAAEoAAAAVmNtYXNlE+dkAAABlAAADxnbHlmg4w
eAgAAAdwAAAhQaGVhZBfh57sAAADQAAAAANmhoZWEIVQQGAAAArAAAACRobXR4FAAAAAAAYAAAAA
UbG9jYQNeBi4AAAHQAAAAADG1heHABGAFgAAABCAAAACBuYW1lH65UOQAACiWAAALNcG9zdFsyKlE
AAAz8AAAAUgABAAAEAAAAAFWEAAAAAAD+AAABAAAAAAAAAAAAAAAAABQABAAAAAQAA7pb8lF8
PPPUACwQAAAAAANpY0WMAAAAAA2ljRyWAAAAAD+AP4AAAAACAACAAAAAAAAAAAAEAAAFVQACQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwQAAAAAXAQAAAAAAAABAAAAAABAAAAAQAAAAA
EAAAABAAAAAQAAAAAACAaaaaawAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEAAgADAAQAAAAAMwCBgKSBCgABAAAAAAD+AP4ACEAQwBlAKkAAAEfBw8HIS8HPwc
lHwcPByEvBz8HJR8HDwchLwc/BycRHw8hPw8RLw8hDw4CXgcGBQUEAwEBAQEDBAUFBgf+hgYGBQU
EAWEBQEDBAUFBgYCOAYGBQUEAwEBAQEDBAUFBg9yAYGBQUEAwEBAQEDBAUFBgYCOAYGBQUEAwE
BAQEDBAUFBg9yAYGBQUEAwEBAQEDBAUFBgbcAQIDBQUHCAkKCgsMDQ0ODQLgDQ4NDQwLCgoJCac
FBQMCAQECAwUFBwgJCgoLDA0NDg39IA0ODQ0MCwoKCQgHBQUdAgFDAQEDBAUFBgYHBgUFBAMBAQE
BAwQFBQYHBgYFBQQDAQG9AQEDBAUFBgCGBgUFBAMBAQEBAwQFBQYGBwYFBQQDAQG9AQEDBAUFBgY
HBgUFBAMBAQEBAwQFBQYHBgYFBQQDAQGz/SANDg0NDAsKCgkIBwUFAwIBAQIDBQUHCAkKCgsMDQ0
ODQLgDQ4NDQwLCgoJCacFBQMCAQECAwUFBwgJCgoLDA0NDgAABAAAAAAD+AP4AD8AggDUARgAAAE
fBw8PLw41Pw8fBicPDx8PMz8OLxAHNzMfEhUPESsBLx9AT8UJREfDyE/DxEvDyEPDgJlCacGBgQ
CAGEBAGMEBQCCHCAkJCwsMDAwNDgwNDAsLCgkICAYFAwMBAQMDBQUHBwgJCQoLCwwMDA4MDAwLCgq
EDg8PDw4PDw8VFBQUEXMTehUWFhYXFxgYehMSERISEREUEBEREBESERkZGRgXFxcXEA8QEBARERE
WFXyVfHhUWFhIeFASXGBkYGRkYGSATEXISEhIRBQMBAGICHBkaGhscGx0UEXMTExMTExoUFRQVFBu
VHBoaGhkYGRkEAgIDGBQVFhYXFxcREREREQEREQ8ODv4aAQIDBQUHCAkKCgsMDQ0ODQLgDQ4NDQw
LCgoJCacFBQMCAQECAwUFBwgJCgoLDA0NDg39IA0ODQ0MCwoKCQgHBQUdAgJXCQoKCwsMDAwNDAw
MCgsJCQgHBgUEAwIBAQIDBQUHCAkKCgsMCw0MDQwLDAoLCQkJBwcGBQQCAGEBAGMEBQYIWQMEBQY
GBwgJDg4PERETExUYFxEhAPDgkIBwUFAwEBAgIEBQYHCA0QEBMUfHcaEREQDw8NDQ0PDQsJCAY
EAWEBMAIEBggJDA4PFg8PERESFBQBwYGBgUEIBsZFhUTERAJCAYGBAMCAGQFBggJChAREhUWGB0
eCAUFBAyHGxcVFBMREQ8KCQgHBgYEBAMCAYT9IA0ODQ0MCwoKCQgHBQUdAgEBAgMFBQcICQoKCww
NDQ4NAuANDg0NDAsKCgkIBwUFAwIBAQIDBQUHCAkKCgsMDQ0OAAIAAAAAA/gD+AArAG8AAAEfAhU
PAwEPay8INT8GMx8DAT8DHwILER8PIT8PES8PIQ8OAvMEAwIBAQME/r8FBQYGBgYFBXkEAWEBAGM
EBQUGBgYGBgViAsOFBgYGBgYF/RoBAGMFBgQICQoKCwwNDQ4NAuANDg0NDAsKCgkIBwUFAwIBAQI
DBQUHCAkKCgsMDQ0ODf0gDQ4NDQwLCgoJCacFBQMCArQFBgYGBgYFBf7FBAMBAQEBAwR2BQUGBgY
GBgUEAwEBAGMEYAE1BAMBAQEBA7j9IA0ODQ0MCwoKCQgHBQUdAgEBAgMFBQcICQoKCwwNDQ4NAuA
NDg0NDAsKCgkIBwUFAwIBAQIDBQUHCAkKCgsMDQ0OAAAJAAAAAAP4A/gAIQBDAGUAhwCpAMsA7QE
PAVMAAAEVDwcvBzU/Bx8GNx8EDwYrAS8GPQE/BtsBHwEFHwMPBysBLwU9AT8GOWEfASUfBw8HIy8
HPwchHwcPByMvBz8HJR8DDwcrAS8FPQE/BjsBHwEFHwMdAQ8FKwEvBz8GOWEfASUVDwcvBzU/Bx8
GJREfDyE/DxEvDyEPDgIgAQIDBAQGBgYGBgYEBAMCAQECAwQEBgYGBgYGBAQDAopiBAMCAQECAwQ
FBQYGBgYFBWIEAwICAwQFBQYGBgYF/t8EAWIBAQIDBGI FBQYGBgYFBQQDAgIDBGI FBQYGBgYFAdw
HBgUFBAMBAQEBAwQFBQYHigYGBgQEAWIBAQIDBAQGBgb+YAYGBgQEAWIBAQIDBAQGBgaKBwYFBQQ
```

```

DAQEBAQMEBQUGBwJlBAMCAQECaWRiBQUGBgYGBQUEAwICaWRiBQUGBgYGBf4bYgQDAgIDBAUFBgY
GBgUFYgQDAgEBaGMEBQUGBgYGBQEEAQIDBAQGBgYGBgYEBAMCAQECaWQEBgYGBgYGBAQDAv3pAQI
DBQUHCAkKCgsMDQ00DQLgDQ4NDQwLCgoJCAcFBQMCAQECaWUFBwgJCgoLDA0NDg39IA00DQ0MCwo
KCQgHBQUdAgEwigcGBQUEAwEBAQEDBAUFBgeKBgYGBAQDAgEBaGMEBAYGTWIFBQYGBgYFBQQDAgI
DBGIFBQYGBgYFBQQDAgIDBAUFBgYGBgUFYgQDAgIDBAUFBgYGBgUFYgQDAgIDmQECaWQEBgYGBgY
GBAQDAgEBaGMEBAYGBgYGBgQEAWIBAQIDBAQGBgYGBgYEBAMCAQECaWQEBgYGBgYGBAQDAgHrBQU
GBgYGBQViBAMCAgMEBQUGBgYGBQViBAMCAgMEYgUFBgYGBgYUFBAMCAgMEYgUFBgYGBgYUFBAMCAgN
LigYGBgQEAWIBAQIDBAQGBgaKBwYFBQQDAQEBAQMEBQUGD/0gDQ4NDQwLCgoJCAcFBQMCAQECaWU
FBwgJCgoLDA0NDg0C4A00DQ0MCwoKCQgHBQUdAgEBaGMFBQcICQoKCwWNDQ4AAAAAEgDeAAEAAAA
AAAAAAQAAAAEAAAAAAAAEAFQABAAEAAAAAAAAIABwAWAAEAAAAAAAAAMAFQAdAAEAAAAAAAAQAFQAYAAE
AAAAAAUACwBHAAEAAAAAAAAAYAFQBSAAEAAAAAAAAoALABnAAEAAAAAAAAsAEgCTAAMAAQQJAAAAAgC
LAAMAAQQJAAEAkGcNaAMAAQQJAAIADgDRAAMAAQQJAAAMAKgDfaAMAAQQJAAQAKgEJAAMAAQQJAAU
AFgEzAAMAAQQJAAAYAKgFJAAMAAQQJAAoAWAFzAAMAAQQJAAAsAJAHLIEthbmJhbiBwcmlvcml0eSB
pY29uc1JlZ3VsYXJlYW5iYW4gcHJpb3JpdHkgaWNvbnNlYW5iYW4gcHJpb3JpdHkgaWNvbnNlZW5i
ZaW9uIDEuMEthbmJhbiBwcmlvcml0eSBpY29uc0ZvbG9uZ2Vuc2Vuc2Vuc2Vuc2Vuc2Vuc2Vuc2Vuc2
pb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmdXNpb24uY29tACAASwBhAG4AYgBhAG4AIABwAHIAaQB
vAHIAaQB0AHkAIABpAGMABwBuAHMAUgBlAGcAdQBsAGEAcgBLAGEAbgBiAGEAbgAgAHAacgBpAG8
AcgBpAHQAeQAgAGkAYwBvAG4AcwBLAGEAbgBiAGEAbgAgAHAacgBpAG8AcgBpAHQAeQAgAGkAYwB
vAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAASwBhAG4AYgBhAG4AIABwAHIAaQBvAHIAaQB0AHk
AIABpAGMABwBuAHMARgBvAG4AdAAgAGcAZQBwAGUAcgBhAHQAeQAgACAAdQBsAGkAbgBnACAAUwB
5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdABlAGQAaQBvAHcAdwB3AC4AcwB5AG4
AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ABQECAQMBBAEFAQYACFRvZG9saXN0B1JldmllldwldB21wbGV0ZWQIUHJvZ3Jlc3MAAAAA)
format('trueType');
        font-weight: normal;
        font-style: normal;
    }
    [class^="sf-icon-"],
    [class*=" sf-icon-"] {
        font-family: 'Kanban priority icons' !important;
        speak: none;
        font-size: 55px;
        font-style: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        line-height: 1;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
    }
</style>

<div id="container">
    <div class="content-wrapper">
        <div id="Kanban"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Toggle columns

Kanban allows to expand or collapse its columns using `allowToggle` inside the `columns` property. When enable the property, it will render the expand or collapse icon to the column header.

By default, collapsed column width is set to `50px`.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open', allowToggle: true },
    { headerText: 'In Progress', keyField: 'InProgress', allowToggle:
true },
    { headerText: 'Testing', keyField: 'Testing', allowToggle: true },
    { headerText: 'Done', keyField: 'Close', allowToggle: true }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Toggle Column</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Toggle Column">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Initially collapsed column

By default, all columns are on expanded state when loading the Kanban board initially. But, you can render the columns with collapsed state using the `isExpanded` property.

The `isExpanded` property only works when enabling the `allowToggle` property on particular column.

In the following example, the backlog column is collapsed on initialization of Kanban board.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open', allowToggle: true,
isExpanded: false },
        { headerText: 'In Progress', keyField: 'InProgress', allowToggle:
true },
        { headerText: 'Testing', keyField: 'Testing', allowToggle: true,
isExpanded: false },
        { headerText: 'Done', keyField: 'Close', allowToggle: true }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Rendering Column with Collapsed State</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Kanban rendering column with collapsed
state">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stacked headers

Stacked headers are the additional headers to column header that will group the similar columns.

Define the grouping of columns **key** value to the **keyFields** property and provide the custom header text name to grouped columns using the **text** property, which is placed inside the **stackedHeaders** property.

In the following code, the kanban columns 'InProgress, Review' are grouped under 'Development Phase' category.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Review', keyField: 'Review' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    stackedHeaders: [
        { text: 'To Do', keyFields: 'Open' },
        { text: 'Development Phase', keyFields: 'InProgress, Review' },
        { text: 'Done', keyFields: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Stacked Headers</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban stacked headers">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
```

```
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Cards in EJ2 JavaScript Kanban control

The cards are main elements in Kanban board, which represent the task information with header and content. The header and content of a card is fetched from the corresponding mapping fields. The card layout can be customized with template also.

Drag-and-drop

Transit or change the card position using the drag-and-drop functionality. By default, the `allowDragAndDrop` property is enabled on the Kanban board, which is used to change the card position by column-to-column or within the column.

Added dotted border on Kanban cells except the dragged clone cells when dragging, which indicates the possible ways for dropping the cards into the cells.

Header

The card header is achieved by mapping the `headerField` property, which is placed inside the `cardSettings` property. By default, the `showHeader` property enabled by Kanban board that is used to show the header at the top of the card.

The `headerField` property of `cardSettings` is mandatory to render the cards in the Kanban board. It acts as a unique field that is used to avoid the duplication of card data. You can not change the `headerField` of mapped data value using the `updateCard` public method or server-side update of data.

In the following demo, the `showHeader` property is disabled on Kanban board.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
```



```

        showHeader: false,
        contentField: 'Summary',
        headerField: 'Id'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Card without Header</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban card without header">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Content

The card's content is fetched from data source using the `contentField` property, which is placed inside the `cardSettings` property. If the `contentField` property is not used, card is rendered with empty content.

Template

You can customize the default card layout using template as per your application needs. This can be achieved by template of the `cardSettings` property.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id',
        template: '#cardTemplate'
    }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Card Template</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban card template">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-kanban/styles/material.css" rel="stylesheet">
```

```

<script id="cardTemplate" type="text/x-jsrender">
  <div class='e-card-content'>
    <table class="card-template-wrap">
      <tbody>
        <tr>
          <td class="CardHeader">Id:</td>
          <td>${Id}</td>
        </tr>
        <tr>
          <td class="CardHeader">Type:</td>
          <td>${Type}</td>
        </tr>
        <tr>
          <td class="CardHeader">Priority:</td>
          <td>${Priority}</td>
        </tr>
        <tr>
          <td class="CardHeader">Summary:</td>
          <td>${Summary}</td>
        </tr>
      </tbody>
    </table>
  </div>
</script>
<style type="text/css">
  .prop-grid .row {
    padding: 10px 0;
  }
  .e-kanban .card-template-wrap td {
    background: none !important;
  }
  .e-kanban .card-template-wrap .CardHeader {
    font-weight: 500;
  }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection

Kanban board allows to select single and multiple selection of cards when mouse or keyboard interactions using `selectionType` property. The property contains following types.

- **None:** No cards are allowed to select from Kanban board.
- **Single:** Only one card allowed to select at a time in the Kanban board.
- **Multiple:** Multiple cards are allowed to select in a board.

Multiple Selection

Select the multiple cards randomly using Ctrl + mouse click and select the multiple cards continuously using Shift + mouse click action on Kanban board. Set **Multiple** in `selectionType` to enable the multiple selection in a board.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id',
    selectionType: 'Multiple'
  }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Multiple Cards Selection</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban multiple cards selection">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Data binding in EJ2 JavaScript Kanban control

The Kanban uses **DataManager**, which supports both RESTful data service binding and JavaScript object array binding. The [dataSource](#) property of Kanban can be assigned either with the instance of **DataManager** or JavaScript object array collection, as it supports the following two data binding methods:

- Local data
- Remote data

Local data

To bind local JSON data to the Kanban, you can simply assign a JavaScript object array to the [dataSource](#) property. The JSON object data source can also be provided as an instance of **DataManager** and assigned to the Kanban **dataSource** property.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',

```

```

        columns: [
            { headerText: 'Backlog', keyField: 'Open' },
            { headerText: 'In Progress', keyField: 'InProgress' },
            { headerText: 'Testing', keyField: 'Testing' },
            { headerText: 'Done', keyField: 'Close' }
        ],
        cardSettings: {
            contentField: 'Summary',
            headerField: 'Id'
        }
    });
    kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Local Data</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban Local Data">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, **DataManager** uses **JsonAdaptor** for binding local data.

Remote data

To bind remote data to kanban component, assign service data as an instance of [DataManager](#) to the [dataSource](#) property. To interact with remote data source, provide the endpoint [url](#).

INDEX.TS

```

import { Kanban, DialogEventArgs } from '@syncfusion/ej2-kanban';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
    adaptor: new ODataAdaptor
});
let kanbanObj: Kanban = new Kanban({
    dataSource: data,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    allowDragAndDrop: false,
    dialogOpen: dialogOpen
});
kanbanObj.appendTo('#Kanban');
function dialogOpen(args: DialogEventArgs): void {
    args.cancel = true;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Remote Data</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban Remote Data">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, [DataManager](#) uses **ODataAdaptor** for remote data-binding.

OData services

[OData](#) is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the DataManager. Refer to the following code example for remote Data binding using OData service.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
    adaptor: new ODataAdaptor,
    crossDomain: true
});
let kanbanObj: Kanban = new Kanban({
    dataSource: data,
    keyField: 'Status',
    columns: [

```



```

        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    allowDragAndDrop: false,
    dialogOpen: dialogOpen
});
kanbanObj.appendTo('#Kanban');
function dialogOpen(args: DialogEventArgs): void {
    args.cancel = true;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban OData</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban OData">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

OData v4 services

The ODataV4 is an improved version of OData protocols, and the [DataManager](#) can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the [odata documentation](#). To bind OData v4 service, use the **ODataV4Adaptor**.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://services.odata.org/v4/northwind/northwind.svc/Suppliers',
    adaptor: new ODataV4Adaptor
});
let kanbanObj: Kanban = new Kanban({
    dataSource: data,
    keyField: 'ContactTitle',
    columns: [
        { headerText: 'Order Administrator', keyField: 'Order Administrator' },
        { headerText: 'Sales Representative', keyField: 'Sales Representative' },
        { headerText: 'Export Administrator', keyField: 'Export Administrator' }
    ],
    cardSettings: {
        contentField: 'ContactName',
        headerField: 'SupplierID'
    },
    allowDragAndDrop: false,
    dialogOpen: dialogOpen
});
kanbanObj.appendTo('#Kanban');
function dialogOpen(args: DialogEventArgs): void {
    args.cancel = true;
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>Kanban ODataV4</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Kanban ODataV4">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Web API

You can use **WebApiAdaptor** to bind kanban with Web API created using OData endpoint.

```
`ts
```

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';

let data: DataManager = new DataManager({
    url: '/api/Tasks',
    adaptor: new WebApiAdaptor
});

let kanbanObj: Kanban = new Kanban({

```

```

dataSource: data,
keyField: 'Status',
columns: [
{ headerText: 'Backlog', keyField: 'Open' },
{ headerText: 'In Progress', keyField: 'InProgress' },
{ headerText: 'Testing', keyField: 'Testing' },
{ headerText: 'Done', keyField: 'Close' }
],
cardSettings: {
contentField: 'Summary',
headerField: 'Id'
},
allowDragAndDrop: false,
dialogOpen: dialogOpen
});
kanbanObj.appendTo('#Kanban');
function dialogOpen(args: DialogEventArgs): void {
args.cancel = true;
}

```

Below server-side controller code to get the Kanban data.

```

`ts
[HttpGet]
public object Get()
{
var data = OrdersDetails.GetAllRecords().ToList();
return data;
}

```

URL adaptor

The CRUD (Create, Read, Update and Delete) actions can be performed easily on Kanban cards using the various adaptors available within the **DataManager**. Most preferably, we will be using **UrlAdaptor** for performing CRUD actions on Kanban.

The CRUD operation in Kanban can be mapped to server-side controller actions using the properties `insertUrl`, `removeUrl`, `updateUrl`, and `crudUrl`.

- `insertUrl` – You can perform a single insertion operation on the server-side.
- `updateUrl` – You can update single data on the server-side.
- `removeUrl` – You can remove single data on the server-side.
- `crudUrl` – You can perform bulk data operation on the server-side.

```
`ts
```

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';

let data: DataManager = new DataManager({
  url: 'Home/DataSource',
  updateUrl: 'Home/Update',
  insertUrl: 'Home/Insert',
  removeUrl: 'Home/Delete',
  adaptor: new UrlAdaptor(),
  crossDomain: true
});

let kanbanObj: Kanban = new Kanban({
  dataSource: data,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  }
});

kanbanObj.appendTo('#Kanban');
```

The server-side controller code to handle the CRUD operations are as follows.

```
`ts
private NORTHWNDEntities db = new NORTHWNDEntities();
public ActionResult DataSource() {
    var DataSource = db.Tasks.ToList();
    return Json(DataSource, JsonRequestBehavior.AllowGet);
}
public ActionResult Insert(Params value) {
    //Insert card data into the database
    return Json(value, JsonRequestBehavior.AllowGet);
}
public ActionResult Update(Params value) {
    //Update card data into the database
    return Json(value, JsonRequestBehavior.AllowGet);
}
public void Delete(Params value) {
    //Delete card data from the database
}

public class Params {
    public int Id { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Assignee { get; set; }
}
`
```

The `crudUrl` is used to update the bulk data sent to the server-side. Multiple selections and `sortBy` as `Index` properties are used for `crudUrl` properties to update the modified bulk data to the server-side.

Custom adaptor

It is possible to create your own custom adaptor by extending the built-in available adaptors. The following example demonstrates the custom adaptor usage and how to add a custom field `TaskId` for the cards by overriding the built-in response processing using the `processResponse` method of the `ODataAdaptor`.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
```

```

class TaskIdAdaptor extends ODataAdapter {
  processResponse(): Object {
    let i: number = 0;
    // calling base class processResponse function
    let original: Object[] = super.processResponse.apply(this,
arguments);
    // adding Task Id
    original.forEach((item: { [key: string]: Object }) => item['Id'] =
'Task - ' + ++i);
    return original;
  }
}
let data: DataManager = new DataManager({
  url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
  adaptor: new TaskIdAdaptor
});
let kanbanObj: Kanban = new Kanban({
  dataSource: data,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  },
  allowDragAndDrop: false,
  dialogOpen: dialogOpen
});
kanbanObj.appendTo('#Kanban');
function dialogOpen(args: DialogEventArgs): void {
  args.cancel = true;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Custom Adaptor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban Custom Adaptor">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sending additional parameters to the server

To add a custom parameter to the data request, use the **addParams** method of **Query** class. Assign the **Query** object with additional parameters to the kanban [query](#) property.

INDEX.TS

```

import { Kanban, DialogEventArgs } from '@syncfusion/ej2-kanban';
import { DataManager, ODataAdaptor, Query } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/Kanban',
    adaptor: new ODataAdaptor
});
let kanbanObj: Kanban = new Kanban({
    dataSource: data,
    query: new Query().addParams('ej2kanban', 'true'),
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',

```



```

        headerField: 'Id'
    },
    allowDragAndDrop: false,
    dialogOpen: dialogOpen
});
kanbanObj.appendTo('#Kanban');
function dialogOpen(args: DialogEventArgs): void {
    args.cancel = true;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Additional Parameter</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban Additional Parameter">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

The parameters added using the [query](#) property will be sent along with the data request for every kanban action.

Handling HTTP error

During server interaction from the kanban, some server-side exceptions may occur, and you can acquire those error messages or exception details

in client-side using the [actionFailure](#) event.

The argument passed to the [actionFailure](#) event contains the error details returned from the server.

INDEX.TS

```
import { Kanban, DialogEventArgs } from '@syncfusion/ej2-kanban';
import { DataManager } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'http://some.com/invalidUrl'
});
let kanbanObj: Kanban = new Kanban({
    dataSource: data,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    actionFailure: (e) => {
        let span: HTMLElement = document.createElement('span');
        kanbanObj.element.parentNode.insertBefore(span, kanbanObj.element);
        span.style.color = '#FF0000';
        span.innerHTML = 'Server exception: 404 Not found';
    }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Action Failure</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban Action Failure">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Server exception: 404 Not found

The [actionFailure](#) event will be triggered not only for the server errors, but also when there is an exception while processing the kanban actions.

Loading data via ajax

You can use Kanban [dataSource](#) property to bind the datasource to Kanban from external ajax request. In the following code, we have fetched the datasource from the server using ajax request and provided that to the [dataSource](#) property by using the **onSuccess** event of ajax.

INDEX.TS

```

import { Kanban, DialogEventArgs } from '@syncfusion/ej2-kanban';
import { Ajax } from '@syncfusion/ej2-base';
let kanbanObj: Kanban = new Kanban({
    keyField: 'ShipCountry',
    columns: [
        { headerText: 'Denmark', keyField: 'Denmark' },
        { headerText: 'Brazil', keyField: 'Brazil' },
        { headerText: 'Switzerland', keyField: 'Switzerland' },
        { headerText: 'Germany', keyField: 'Germany' }
    ],
    cardSettings: {
        contentField: 'ShippedDate',
        headerField: 'OrderID'
    }
});
kanbanObj.appendTo('#Kanban');
let button = document.getElementById('btn');
button.addEventListener("click", function(e) {
    let ajax = new Ajax("https://ej2services.syncfusion.com/production/web-services/api/Orders", "GET");
    ajax.send();
    ajax.onSuccess = function (data: string) {
        kanbanObj.dataSource = JSON.parse(data);
    };
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban AJAX Binding</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban AJAX Binding">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <button id="btn" class="e-btn">Click to load Kanban
data</button>
            <div id="Kanban"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server-side crud actions.

Swimlane in EJ2 JavaScript Kanban control

Swimlanes are horizontal categorizations of cards on the Kanban board. It is used for grouping of cards, which brings transparency to the workflow process.

Render swimlane row

Cards can be grouped based on **keyField** and displayed in rows, which are separated by columns. It is mandatory to define the **keyField** that is mapped from the datasource for rendering swimlane rows in the Kanban board.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    swimlaneSettings: {
        keyField: 'Assignee'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Swimlane</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban board with swimlane">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom row text

Customize the swimlane row header text by using the `textField` property mapped from datasource.

It is not mandatory to define the `textField` to `swimlaneSettings`. It will automatically consider the `keyField` to swimlane row header text.

If the mapping `textField` key is not present in the datasource, it will consider the swimlane `keyField` as swimlane row header text.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    swimlaneSettings: {
        keyField: 'Assignee',
        textField: 'AssigneeName'
    }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Swimlane Custom Header Row Text</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Swimlane custom row text">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Template

You can customize the Kanban swimlane row by using the `template` property, which is specified within the `swimlaneSettings` property. In this demo, the swimlane header is customized with HTML element.

You can get the following swimlane data when using a swimlane template.

- `keyField`
- `textField`
- `count`

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    swimlaneSettings: {
        keyField: 'Assignee',
        textField: 'AssigneeName',
        template: '#swimlaneTemplate'
    }
});
kanbanObj.appendTo('#Kanban');

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Swimlane Template</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Swimlane template">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script id="swimlaneTemplate" type="text/x-jsrender">
    <div class='swimlane-template e-swimlane-template-table'>
      <div class="e-swimlane-row-text"><span>${textField}</span></div>
    </div>
  </script>
  <style>
    .swimlane-template {
      font-size: 15px;
      font-weight: 500;
    }
    .swimlane-template img {
      height: 24px;
      width: 24px;
      border-radius: 50%;
    }
    .swimlane-template span {
      padding-left: 10px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div class="content-wrapper">
    <div id="Kanban"></div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sorting

Swimlane rows are rendered on descending order when using the `sortBy` property set to `Descending` order. By default, swimlane rows are rendered by **Ascending** order.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  },
  swimlaneSettings: {
    keyField: 'Assignee',
    sortBy: 'Descending'
  }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Swimlane Sorting</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Swimlane sorting">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag-and-drop

By default, The Kanban does not allow dragging the cards across the swimlane rows. Enabling the `dragAndDrop` property allows you to drag the cards across the swimlane rows, which is specified inside `swimlaneSettings` property.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',

```

```

        headerField: 'Id'
    },
    swimlaneSettings: {
        keyField: 'Assignee',
        allowDragAndDrop: true
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Across Swimlane Drag and Drop</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Swimlane drag and drop">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Create empty row

You can render the empty swimlane row by enabling the `showEmptyRow` property. If mapping `keyField` does not have cards, empty swimlane row will be rendered.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  },
  swimlaneSettings: {
    keyField: 'Assignee',
    showEmptyRow: true
  }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Swimlane Row with Empty Rows</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Empty swimlane row">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Calculate cards count

Users can show or hide the cards count by swimlane row in header when enabling the `showItemCount` property, which is enabled by default on the Kanban board.

Provided localization support for **items** text.

In below demo, disabled on `showItemCount` property on rendering swimlane row without total count.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    swimlaneSettings: {
        keyField: 'Assignee',
        showItemCount: false
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Swimlane Cards Count</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Swimlane row cards count">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable frozen rows

Frozen rows provide an option to make the current swimlane row header text always visible on top of the content while scrolling the Kanban content. The swimlane header text will be changed dynamically, when you scroll to another swimlane row.

By default, the `enableFrozenRows` property is set as `false`. If you wish to show the swimlane frozen rows, you can enable the `enableFrozenRows` property.

This feature support only when using Kanban content scrolling.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    height: 500,
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    swimlaneSettings: {
        keyField: 'Assignee',
        enableFrozenRows: true
    }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Swimlane Cards Count</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Swimlane row cards count">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-kanban/styles/material.css" rel="stylesheet">
```



```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Drag and drop in EJ2 JavaScript Kanban control

All cards can be dragged and dropped across the columns or within the columns or swimlane row or kanban to an external source and vice versa.

The following drag and drop types are available in the Kanban board.

- Internal drag and drop
- Column drag and drop
- Swimlane drag and drop
- External drag and drop
- Kanban to Kanban
- Kanban to External source and vice versa.

Dropped card position varies based on the `sortSettings` property.

Internal drag and drop

Allows the user to drag and drop the cards within the kanban board. Based on this, we can categorize into two ways.

- Column drag and drop
- Swimlane drag and drop

Column drag and drop

By default, all cards can be dragged and dropped across the columns and within the columns. You cannot drag and drop the cards when disabling the `allowDragAndDrop` property.

You can prevent the drag or drop behavior of the particular column by disabling the `allowDrag` or `allowDrop` property.

You can also control the flow of transition cards between the columns by using the `transitionColumns` property.

In the following example, disable the drag and drop behavior on the Kanban board.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    allowDragAndDrop: false
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Across Swimlane Drag and Drop</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Swimlane drag and drop">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Swimlane drag and drop

By default, Swimlane allows drag and drop across the columns within the swimlane row. Kanban does not allow dragging the cards across the swimlane rows.

Enabling the `dragAndDrop` property allows you to drag the cards across the swimlane rows, which is specified inside the `swimlaneSettings` property.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    swimlaneSettings: {
        keyField: 'Assignee',
        allowDragAndDrop: true
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Across Swimlane Drag and Drop</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Swimlane drag and drop">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

External drag and drop

Allows the user to drag and drop the cards from one kanban to another kanban or Kanban to an external source and vice versa.

Kanban to kanban

Drag and drop the card from one kanban to another kanban and vice versa. This can be achieved by specifying the `externalDropId` property which is used to specify the id of the dropped kanban element and the `dragStop` event which is used to delete the card on dragged Kanban and add the card on dropped Kanban using the `deleteCard` and `addCard` public methods.

Before adding a card to dropped kanban, you can manually change the card data `headerField` when the same card data `headerField` is dropped to another Kanban.

In the following example, Drag the card from one Kanban and drop it into another kanban using the `dragStop` event. In this event, remove the card from the dragged Kanban by using the `deleteCard` public method and add the card to the dropped Kanban by using the `addCard` public method.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { closest } from '@syncfusion/ej2-base';
import { kanbanAData, kanbanBData } from './datasource.ts';
const kanbanObjA: Kanban = new Kanban({
  dataSource: kanbanAData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  externalDropId: ['#kanbanB'],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id',
    selectionType: 'Multiple'
  },
  showEmptyColumn: true,
  dragStop: (args: DragEventArgs): void => {
    if (closest(args.event.target as Element, '#kanbanB')) {
      kanbanObjA.deleteCard(args.data);
      args.data.forEach((card: Record<string, any>, i: number) => {
        const index: number =
          kanbanObjB.kanbanData.findIndex((colData: Record<string, any>) =>
            colData[kanbanObjB.cardSettings.headerField] ===
            card[kanbanObjB.cardSettings.headerField]);
        if (index !== -1) {
          card[kanbanObjB.cardSettings.headerField] =
            Math.max(...kanbanObjB.kanbanData.map(
              (obj: Record<string, string>) =>
                parseInt(obj[kanbanObjB.cardSettings.headerField], 10))) + ++i;
        }
      });
      kanbanObjB.addCard(args.data, args.dropIndex);
      args.cancel = true;
    }
  }
});
kanbanObjA.appendTo('#kanbanA');
const kanbanObjB: Kanban = new Kanban({
  dataSource: kanbanBData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
    contentField: 'Summary',
```

```

        headerField: 'Id'
    },
    externalDropId: ['#kanbanA'],
    showEmptyColumn: true,
    dragStop: (args: DragEventArgs): void => {
        if (closest(args.event.target as Element, '#kanbanA')) {
            kanbanObjB.deleteCard(args.data);
            args.data.forEach((card: Record<string, any>, i: number) => {
                const index: number =
kanbanObjA.kanbanData.findIndex((colData: Record<string, any>) =>
                    colData[kanbanObjA.cardSettings.headerField] ===
card[kanbanObjA.cardSettings.headerField]);
                if (index !== -1) {
                    card[kanbanObjA.cardSettings.headerField] =
Math.max(...kanbanObjA.kanbanData.map(
                        (obj: Record<string, string>) =>
parseInt(obj[kanbanObjA.cardSettings.headerField], 10))) + ++i;
                }
            });
            kanbanObjA.addCard(args.data, args.dropIndex);
            args.cancel = true;
        }
    }
});
kanbanObjB.appendTo('#kanbanB');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban to Kanban Drag and Drop</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban to Kanban drag and drop">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="container-fluid">
            <div class="row">
                <div class="col-sm-6">
                    <h4>Kanban A</h4>
                    <div id="kanbanA"></div>
                </div>
                <div class="col-sm-6">
                    <h4>Kanban B</h4>
                    <div id="kanbanB"></div>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

Treeview to Kanban

Drag the card from the Kanban board and drop it to the Treeview component and vice versa.

In the following sample, remove the data from the Kanban board using the `deleteCard` public method and add to the Treeview component using the `addNodes` public method at Kanban `dragStop` event when dragging the card and dropping it to the Treeview component. Remove the data from Treeview using the `removeNodes` public method and add to Kanban board using the `openDialog` public method when dragging the list from the Treeview component and dropping it to the kanban board.

INDEX.TS

```

import { Kanban, KanbanModel, DragEventArgs } from '@syncfusion/ej2-kanban';
import { kanbanData, treeViewData } from './datasource.ts';
import { closest } from '@syncfusion/ej2-base';
import { TreeView, DragAndDropEventArgs } from '@syncfusion/ej2-
navigations';
const kanbanOptions: KanbanModel = {
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    externalDropId: ['#TreeView'],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    }
};

```

```

    },
    showEmptyColumn: true,
    dragStop: (args: DragEventArgs): void => {
        if (closest(args.event.target as Element, '#TreeView')) {
            kanbanObj.deleteCard(args.data);
            treeObj.addNodes(args.data);
            args.cancel = true;
        }
    }
};
const kanbanObj: Kanban = new Kanban(kanbanOptions);
kanbanObj.appendTo('#Kanban');
let treeObj: TreeView = new TreeView({
    fields: { dataSource: treeViewData, id: 'Id', text: 'Status' },
    allowDragAndDrop: true,
    nodeDragStop: onTreeDragStop,
    nodeTemplate: '#treeTemplate'
});
treeObj.appendTo('#TreeView');
function onTreeDragStop(args: DragAndDropEventArgs): void {
    if (closest(args.event.target as Element, '#Kanban')) {
        let treeData: { [key: string]: Object }[] =
            treeObj.fields.dataSource as { [key: string]: Object }[];
        const filteredData: { [key: string]: Object }[] =
            treeData.filter((item: any) => item.Id !==
                parseInt(args.draggedNodeData.id as string, 10));
        treeObj.removeNodes([args.draggedNodeData.id] as string[]);
        kanbanObj.openDialog('Add', filteredData[0]);
        args.cancel = true;
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban to TreeView Drag and Drop</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban to TreeView drag and drop">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="container-fluid">
            <div class="row">

```



```

        <div class="col-sm-6">
        <h4>Kanban</h4>
        <div id="Kanban"></div>
        </div>
        <div class="col-sm-6">
        <h4>TreeView</h4>
        <div id="TreeView" style="border: 1px solid
#e0e0e0"></div>
        </div>
        </div>
        </div>
        </div>
        <script id="treeTemplate" type="text/x-template">
        <div id="treelist">
        <div id="treeviewlist">${Id} - ${Status}</div>
        </div>
        </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Schedule to Kanban

Drag the card from the Kanban board and drop it to the Schedule component and vice versa.

In the following sample, remove the data from the Kanban board using the `deleteCard` public method and add to the schedule component using the `addNodes` public method at Kanban `dragStop` event when dragging the card and dropping it to the Treeview component. Remove the data from Treeview using the `removeNodes` public method and add to Kanban board using the `addCard` public method when dragging the list from the Treeview component and dropping it to the kanban board.

INDEX.TS

```

import { Kanban, KanbanModel, DragEventArgs } from '@syncfusion/ej2-kanban';
import { kanbanData, scheduleData } from './datasource.ts';
import { Schedule, TimelineViews, TimelineMonth, Resize, DragAndDrop } from '@syncfusion/ej2-schedule';
import { closest, removeClass } from '@syncfusion/ej2-base';
const kanbanOptions: KanbanModel = {
    dataSource: kanbanData,
    keyField: 'DepartmentName',
    columns: [
        { headerText: 'GENERAL', keyField: 'GENERAL' }
    ],
    externalDropId: ['#Schedule'],
    cardSettings: {
        contentField: 'Name',
        headerField: 'Id',
        selectionType: 'Multiple'
    },
    showEmptyColumn: true,

```

```

dragStop: (args: DragEventArgs): void => {
    if (closest(args.event.target as Element, '#Schedule')) {
        kanbanObj.deleteCard(args.data);
        scheduleObj.openEditor(args.data[0], 'Add', true);
        args.cancel = true;
    }
};

const kanbanObj: Kanban = new Kanban(kanbanOptions);
kanbanObj.appendTo('#Kanban');
Schedule.Inject(TimelineViews, TimelineMonth, Resize, DragAndDrop);
let scheduleObj: Schedule = new Schedule({
    width: '350px',
    height: '650px',
    selectedDate: new Date(2018, 7, 1),
    currentView: 'TimelineDay',
    eventDragArea: "body",
    cssClass: 'schedule-drag-drop',
    workHours: {
        start: '08:00',
        end: '18:00'
    },
    views: [
        { option: 'TimelineDay' },
        { option: 'TimelineMonth' }
    ],
    group: {
        enableCompactView: false,
        resources: ['Departments', 'Consultants']
    },
    resources: [
        {
            field: 'DepartmentID', title: 'Department',
            name: 'Departments', allowMultiple: false,
            dataSource: [
                { Text: 'GENERAL', Id: 1, Color: '#bbdc00' },
                { Text: 'DENTAL', Id: 2, Color: '#9e5fff' }
            ],
            textField: 'Text', idField: 'Id', colorField: 'Color'
        },
        {
            field: 'ConsultantID', title: 'Consultant',
            name: 'Consultants', allowMultiple: false,
            dataSource: [
                { Text: 'Alice', Id: 1, GroupId: 1, Color: '#bbdc00',
Designation: 'Cardiologist' },
                { Text: 'Nancy', Id: 2, GroupId: 2, Color: '#9e5fff',
Designation: 'Orthodontist' },
                { Text: 'Robert', Id: 3, GroupId: 1, Color: '#bbdc00',
Designation: 'Optometrist' },
                { Text: 'Robson', Id: 4, GroupId: 2, Color: '#9e5fff',
Designation: 'Periodontist' },
                { Text: 'Laura', Id: 5, GroupId: 1, Color: '#bbdc00',
Designation: 'Orthopedic' },
                { Text: 'Margaret', Id: 6, GroupId: 2, Color: '#9e5fff',
Designation: 'Endodontist' }
            ],

```

```

        textField: 'Text', idField: 'Id', groupIDField: 'GroupId',
        colorField: 'Color'
    },
    ],
    eventSettings: {
        dataSource: scheduleData,
        fields: {
            subject: { title: 'Patient Name', name: 'Name' },
            startTime: { title: 'From', name: 'StartTime' },
            endTime: { title: 'To', name: 'EndTime' },
            description: { title: 'Reason', name: 'Description' }
        }
    },
    dragStop: scheduleDragStop
});
scheduleObj.appendTo('#Schedule');
function scheduleDragStop(args: ScheduleDragEventArgs): void {
    if (closest(args.event.target as Element, '#Kanban')) {
        scheduleObj.deleteEvent(args.data.Id);
        removeClass([scheduleObj.element.querySelector('.e-selected-cell')],
        'e-selected-cell');
        kanbanObj.openDialog('Add', args.data);
        args.cancel = true;
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban to Schedule Drag and Drop</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban to Schedule drag and drop">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="container-fluid">
            <div class="row">
                <div class="col-sm-6">
                    <h4>Kanban</h4>
                    <div id="Kanban"></div>
                </div>
                <div class="col-sm-6">
                    <h4>Schedule</h4>

```

```

        <div id="Schedule"></div>
      </div>
    </div>
  </div>
  <script id="resource-template" type="text/x-template">
    <div class="template-wrap">
      <div class="specialist-category">
        ${getConsultantImage(data)}
        <div class="specialist-
name">${getConsultantName(data)}</div>
        <div class="specialist-
designation">${getConsultantDesignation(data)}</div>
      </div>
    </div>
  </script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sort in EJ2 JavaScript Kanban control

The Kanban provides built-in support to arrange the cards in their columns based on the JSON data order and drop the cards in the columns based on the dropped clone. Initially, users can change the arrangement of cards in the columns and position of the dropped card by using the [sortBy](#) property. The [sortBy](#) property contains three enumeration values as follows.

- Index
- DataSourceOrder
- Custom

Index

SortBy **Index** property can be used with or without [field](#) mapping.

Index without field mapping

By default, SortBy **Index** property support without any [field](#) mapping. In this behavior, cards are loaded based on the JSON data order and cards are dropped based on the dropped clone.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ]
});

```

```

    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Index without mapping</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban Index without mapping">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Index with field mapping

SortBy **Index** property also supports with **field** mapping. In this behavior, cards are loaded based on mapping **field** values, and cards are dropped based on the dropped clone.

Cards are placed in a particular position in the columns where you can drop the cards by specifying the **field** property, which is mapped from the data source. This property allows the users to drop the cards in the Kanban board where the dropped clone is created exactly. It is also helpful to render the cards based on the **field** property value.

The **field** property mapping key value must be in **number** format.

The following cases will dynamically change their **field** value when dropping the cards.

- If the cell has no cards, the dropped card **field** value does not change.
- If the cell has one card and dropped a card to the last position or previous/next cards that do not have continuous order, then the dropped card **field** value will be changed based on their previous card value.
- If the cell has one card and dropped a card on the previous position, then it will compare both the values, and the dropped card **field** value will be changed if the cards have continuous order otherwise values will not be changed.
- When the previous and next cards do not have continuous order, the dropped card **field** value will be changed based on the previous card value.
- When the previous and next cards have continuous order or odd/even value, then the **field** value of the dropped card and the cards followed by the dropped card will be changed based on the **previous** card value with continuous order.

For Example,

Continuous Order -

Consider, Column A has Card A with priority value **1**, Card B with priority value **2**, and Card C with priority value **3**.

and Column B has Card D with priority value **5**, then the dropped Card D will be placed between Card A and Card B. Now, the Cards D, B, and C will be dynamically changed to the priority values as **2**, **3**, and **4** respectively.

Odd/Even order -

Consider, Column A has Card A with priority value **1**, Card B with priority value **3**, and Card C with priority value **5**.

and Column B has Card D with priority value **5**, then the Dropped Card D will be placed between Card A and Card B. Now, the Cards D, B, and C will be dynamically changed to the priority values as **2**, **3**, and **5** respectively.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
```

```

    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    sortSettings: {
        sortBy: 'Index',
        field: 'RankId'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Index without mapping</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban Index without mapping">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>

```

```

        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

DataSource Order

The `SortBy DataSourceOrder` property does not require any [field](#) mapping. In this behavior, cards are loaded based on the JSON data order, and also cards are dropped based on the JSON data order.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    sortSettings: {
        sortBy: 'DataSourceOrder'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Index without mapping</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban Index without mapping">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom

Custom with field mapping

The SortBy **Custom** property must require datasource [field](#) mapping. In this behavior, cards are loaded based on the [field](#) mapping value and also cards are dropped based on the [field](#) mapping value.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    sortSettings: {

```

```

        sortBy: 'Custom',
        field: 'Summary'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Index without mapping</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban Index without mapping">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Change the direction

Kanban board also provides support for aligning the cards in the columns using the [direction](#) property inside the [sortSettings](#) property. Based on this, cards can be aligned in the columns either in **Ascending** or **Descending** order. Sorting direction will be performed based on [sortBy](#) property.

By default, cards are aligned in the columns based on **Ascending** order.

In the following sample, cards are aligned in **Descending** order.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    sortSettings: {
        sortBy: 'Custom',
        field: 'Summary',
        direction: 'Descending'
    }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Index without mapping</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban Index without mapping">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dialog in EJ2 JavaScript Kanban control

The Kanban provides built-in support to add, edit and delete a card using dialog module. User can edit a card using the following ways.

- Built-in dialog module
- Custom Fields
- Dialog template

Default Dialog

When double-click on the cards, the dialog is opened with below fields to edit a card. This dialog contains **Delete**, **Save** and **Cancel** buttons.

- To edit a card, modify the card details and click the **Save** button.
- To delete a card, click **Delete** button.
- Click on the **Cancel** button to cancel the editing action.

The dialog displays with the following fields which mapped to dialog fields by default.

Key | Type | Text

cardSettings.headerField | Input | ID

keyField | DropDown | -

cardSettings.contentField | TextArea | -

cardSettings.priority(If applicable) | Numeric | -

swimlaneSettings.keyField(If applicable) | DropDown | -

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Cards</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban Cards">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Custom Fields

You can change the default fields of dialog using `fields` property inside the `dialogSettings` property. The `key` property used to map the `dataSource` value and rendered the corresponding component based on specified `type` property.

The following types are available in dialog fields.

- String
- Numeric
- TextArea
- DropDown
- TextBox
- Input

If `type` is not defined in the fields, then it renders as the HTML input element in dialog.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  },
  dialogSettings: {
    fields: [
      { key: 'Id', type: 'Input' },
      { key: 'Status', type: 'DropDown' },
      { key: 'Estimate', type: 'Numeric' },
```

```

        { key: 'Summary', type: 'TextArea' }
    ]
}
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Custom Label Dialog</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban Custom Label Dialog">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom Fields label

By default, the fields **key** mapping value is considered as a **label** and you can change this label by using **text** property.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    dialogSettings: {
        fields: [
            { text: 'ID', key: 'Id', type: 'Input' },
            { key: 'Status', type: 'DropDown' },
            { key: 'Estimate', type: 'Numeric' },
            { key: 'Summary', type: 'TextArea' }
        ]
    }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Label Dialog</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban Label Dialog">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Fields Validation

The dialog fields can be validated while click on the **Save** button. This can be achieved by using **validationRules** property.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    dialogSettings: {
        fields: [
            { text: 'ID', key: 'Id', type: 'Input' },
            { key: 'Status', type: 'DropDown' },
            { key: 'Estimate', type: 'Numeric', validationRules: { range:
[0, 1000] } },
            { key: 'Summary', type: 'TextArea', validationRules: { required:
true } }
        ]
    }
});

```

```

    }
  });
  kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Dialog Fields Validation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban Dialog Fields Validation">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Dialog Template

Using the dialog template, you can render your own dialog by defining the **template** property. Initialize the template as SCRIPT element Id or HTML string which holds the template and map it to the template property.

INDEX.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { NumericTextBox, TextBox } from '@syncfusion/ej2-inputs';
import { Kanban, DialogEventArgs } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    dialogSettings: {
        template: '#dialogTemplate'
    },
    dialogOpen: onDialogOpen
});
kanbanObj.appendTo('#Kanban');
let statusData: string[] = ['Open', 'InProgress', 'Close'];
let assigneeData: string[] = ['Nancy Davloio', 'Andrew Fuller', 'Janet Leverling', 'Steven walker', 'Robert King', 'Margaret hamilt', 'Michael Suyama'];
let priorityData: string[] = ['Low', 'Normal', 'Critical', 'Release Breaker', 'High'];
function onDialogOpen(args: DialogEventArgs): void {
    if (args.requestType !== 'Delete') {
        let curData: { [key: string]: Object } = args.data;
        let filledTextBox: TextBox = new TextBox({});
        filledTextBox.appendTo(args.element.querySelector('#Id') as HTMLInputElement);
        let numericObj: NumericTextBox = new NumericTextBox({
            value: curData.Estimate as number, placeholder: 'Estimate'
        });
        numericObj.appendTo(args.element.querySelector('#Estimate') as HTMLInputElement);
        let statusDropObj: DropDownList = new DropDownList({
            value: curData.Status as string, popupHeight: '300px',
            dataSource: statusData, fields: { text: 'Status', value: 'Status' }, placeholder: 'Status'
        });
        statusDropObj.appendTo(args.element.querySelector('#Status') as HTMLInputElement);
        let assigneeDropObj: DropDownList = new DropDownList({
            value: curData.Assignee as string, popupHeight: '300px',
```

```

        dataSource: assigneeData, fields: { text: 'Assignee', value:
'Assignee' }, placeholder: 'Assignee'
    });
    assigneeDropObj.appendTo(args.element.querySelector('#Assignee')
as HTMLInputElement);
    let priorityObj: DropDownList = new DropDownList({
        value: curData.Priority as string, popupHeight: '300px',
        dataSource: priorityData, fields: { text: 'Priority', value:
'Priority' }, placeholder: 'Priority'
    });
    priorityObj.appendTo(args.element.querySelector('#Priority') as
HTMLInputElement);
    let textareaObj: TextBox = new TextBox({
        placeholder: 'Summary',
        multiline: true
    });
    textareaObj.appendTo(args.element.querySelector('#Summary') as
HTMLInputElement);
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Dialog Template</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban Dialog Template">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script id="dialogTemplate" type="text/x-template">
        <table>
            <tbody>
                <tr>
                    <td class="e-label">ID</td>
                    <td>
                        <input id="Id" name="Id" type="text" class="e-field"
value="{Id}" disabled required/>

```

```

        </td>
      </tr>
      <tr>
        <td class="e-label">Status</td>
        <td>
          <input type="text" name="Status" id="Status"
class="e-field" value=${Status} required />
        </td>
      </tr>
      <tr>
        <td class="e-label">Assignee</td>
        <td>
          <input type="text" name="Assignee" id="Assignee"
class="e-field" value=${Assignee} />
        </td>
      </tr>
      <tr>
        <td class="e-label">Priority</td>
        <td>
          <input type="text" name="Priority" id="Priority"
class="e-field" value=${Priority} />
        </td>
      </tr>
      <tr>
        <td class="e-label">Summary</td>
        <td>
          <textarea type="text" name="Summary" id="Summary"
class="e-field" value=${Summary}>${Summary}</textarea>
          <span class="e-float-line"></span>
        </td>
      </tr>
    </tbody>
  </table>
</script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent Dialog

The Kanban allows to prevent to open a dialog on card double-click by enabling `args.cancel` in `dialogOpen` event.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData, DialogEventArgs } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  },
  dialogOpen: 'dialogOpen'
});
kanbanObj.appendTo('#Kanban');
function dialogOpen(args: DialogEventArgs): void {
  args.cancel = true;
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Prevent Dialog Open</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban Prevent Dialog Open">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Persisting data in server

The modified card data can be persisted in the database using the RESTful web services. All the CRUD operations in the Kanban are done through [DataManager](#). The [DataManager](#) has an option to bind all the CRUD related data in server-side.

For your information, the ODataAdaptor persists data in the server as per OData protocol.

In the below section covers how to get the edited data details on the server-side using the [UrlAdaptor](#).

URL adaptor

You can use the [UrlAdaptor](#) of [DataManager](#) when binding data source for remote data. In the initial load of Kanban, data are fetched from remote data and bound to the Kanban using [url](#) property of [DataManager](#).

You can map the CRUD operation in Kanban can be mapped to server-side controller actions using the properties [insertUrl](#), [removeUrl](#), [updateUrl](#), and [crudUrl](#).

- [insertUrl](#) – You can perform single insertion operation on server-side.
- [updateUrl](#) – You can update single data on server-side.
- [removeUrl](#) – You can remove single data on server-side.
- [crudUrl](#) – You can perform bulk data operation on server-side.

The following code example describes the above behavior.

```
`ts
```

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({

```

```

url: 'Home/DataSource',
updateUrl: 'Home/Update',
insertUrl: 'Home/Insert',
removeUrl: 'Home/Delete',
adaptor: new UrlAdaptor(),
crossDomain: true
});
let kanbanObj: Kanban = new Kanban({
dataSource: data,
keyField: 'Status',
columns: [
{ headerText: 'Backlog', keyField: 'Open' },
{ headerText: 'In Progress', keyField: 'InProgress' },
{ headerText: 'Testing', keyField: 'Testing' },
{ headerText: 'Done', keyField: 'Close' }
],
cardSettings: {
contentField: 'Summary',
headerField: 'Id'
}
});
kanbanObj.appendTo('#Kanban');

```

The server-side controller code to handle the CRUD operations are as follows.

```

`ts
private NORTHWNDEntities db = new NORTHWNDEntities();
public ActionResult DataSource() {
var DataSource = db.Tasks.ToList();
return Json(DataSource, JsonRequestBehavior.AllowGet);
}
public ActionResult Insert(Params value) {
//Insert card data into the database
return Json(value, JsonRequestBehavior.AllowGet);
}

```



```

}
public ActionResult Update(Params value) {
//Update card data into the database
return Json(value, JsonRequestBehavior.AllowGet);
}
public void Delete(Params value) {
//Delete card data from the database
}
public class Params {
public int Id { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Assignee { get; set; }
}
`ts

```

Insert card

Using the `insertUrl` property, you can specify the controller action mapping URL to perform insert operation on the server-side.

The following code example describes the above behavior.

```

`ts
public ActionResult Insert(Params value)
{
//Insert card in the database
}
`ts

```

The newly added record details are bound to the `value` parameter.

Update card

Using the `updateUrl` property, the controller action mapping URL can be specified to perform save/update operation on the server-side.

The following code example describes the above behavior.

```

`ts
public ActionResult Update(Params value)
{
//Update card data in the database
}
`ts

```

```
}
,
```

The updated record details are bound to the `value` parameter.

Delete card

Using the `removeUrl` property, the controller action mapping URL can be specified to perform delete operation on the server-side.

The following code example describes the above behavior.

```
`ts
public void Delete(int key)
{
//Delete card in the database
}
,
```

The deleted card primary key value is bound to the `key` parameter.

CRUD URL

Using the `crudUrl` property, the controller action mapping URL can be specified to perform all the CRUD operations at the server-side using a single method instead of specifying a separate controller action method for CRUD (insert, update and delete) operations.

The action parameter of `crudUrl` is used to get the corresponding CRUD action.

The following code example describes the above behavior.

```
`ts
import { Kanban } from '@syncfusion/ej2-kanban';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
url: 'Home/DataSource',
updateUrl: 'Home/UpdateData',
insertUrl: 'Home/UpdateData',
removeUrl: 'Home/UpdateData',
crudUrl: 'Home/UpdateData',
adaptor: new UrlAdaptor(),
crossDomain: true
});
let kanbanObj: Kanban = new Kanban({
dataSource: data,
```

```

keyField: 'Status',
columns: [
{ headerText: 'Backlog', keyField: 'Open' },
{ headerText: 'In Progress', keyField: 'InProgress' },
{ headerText: 'Testing', keyField: 'Testing' },
{ headerText: 'Done', keyField: 'Close' }
],
cardSettings: {
contentField: 'Summary',
headerField: 'Id'
}
});
kanbanObj.appendTo('#Kanban');
`ts
private NORTHWNDEntities db = new NORTHWNDEntities();
public ActionResult DataSource() {
var DataSource = db.Tasks.ToList();
return Json(DataSource, JsonRequestBehavior.AllowGet);
}
public ActionResult UpdateData(EditParams param) {
if (param.action == "insert" || (param.action == "batch" && param.added != null)) {
if (param.action == "insert") {
db.Tasks.Add(param.value);
} else {
foreach (var temp in param.added) {
db.Tasks.Add(temp);
}
}
}
if (param.action == "update" || (param.action == "batch" && param.changed != null)) {
if (param.action == "update") {
Task old = db.Tasks.Where(o => o.Id == param.value.Id).SingleOrDefault();

```

```
if (old != null) {
    db.Entry(old).CurrentValues.SetValues(param.value);
}
} else {
    foreach (var temp in param.changed) {
        Task old = db.Tasks.Where(o => o.Id == temp.Id).SingleOrDefault();
        if (old != null) {
            db.Entry(old).CurrentValues.SetValues(temp);
        }
    }
}

if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) {
    if (param.action == "remove") {
        int key = Convert.ToInt32(param.key);
        db.Tasks.Remove(db.Tasks.Where(o => o.Id == key).SingleOrDefault());
    } else {
        foreach (var temp in param.deleted) {
            db.Tasks.Remove(db.Tasks.Where(o => o.Id == temp.Id).SingleOrDefault());
        }
    }
}

db.SaveChanges();
return Json(param, JsonRequestBehavior.AllowGet);
}

public class EditParams {
    public string key { get; set; }
    public string action { get; set; }
    public List<Tasks> added { get; set; }
    public List<Tasks> changed { get; set; }
    public List<Tasks> deleted { get; set; }
    public Tasks value { get; set; }
}
```

The `crudUrl` is used to update the bulk data sent to the server-side. Multiple selections and `sortBy` as `Index` properties are used for `crudUrl` properties to update the modified bulk data to the server-side.

Tooltip in EJ2 JavaScript Kanban control

The tooltip is used to show the card information when the cursor hover over the card elements using the `enableTooltip` property. Tooltip content is dynamically set based on hovering over the card elements.

If you wish to show tooltip on Kanban board custom elements, you need to add `e-tooltip-text` class name of a particular element.

Tooltip template

You can customize the tooltip content with any HTML or CSS element and styling using the `tooltipTemplate` property. In the following demo, the tooltip is customized with HTML elements.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  },
  enableTooltip: true,
  tooltipTemplate: '#tooltipTemplate'
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Tooltip Template</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Tooltip template">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script id="tooltipTemplate" type="text/x-template">
  <div class='e-kanbanTooltipTemp'>
    <table>
      <tr>
        <td class="details">
          <table>
            <colgroup>
              <col style="width:30%">
              <col style="width:70%">
            </colgroup>
            <tbody>
              <tr>
                <td class="CardHeader">Assignee:</td>
                <td>${Assignee}</td>
              </tr>
              <tr>
                <td class="CardHeader">Type:</td>
                <td>${Type}</td>
              </tr>
              <tr>
                <td class="CardHeader">Estimate:</td>
                <td>${Estimate}</td>
              </tr>
              <tr>
                <td class="CardHeader">Summary:</td>
                <td>${Summary}</td>
              </tr>
            </tbody>
          </table>
        </td>
      </tr>
    </table>
  </div>
</script>
<style type="text/css">
  .e-kanbanTooltipTemp {
    width: 250px;
    padding: 3px;
  }
  .e-kanbanTooltipTemp>table {
    width: 100%;
  }
  .e-kanbanTooltipTemp td {
    vertical-align: top;
  }

```

```

</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Validation in EJ2 JavaScript Kanban control

Validate particular column using the **minCount** or **maxCount** properties. The corresponding columns gets different appearance when validation fails. In default layout, **constraintType** property accept only **Column** type. In swimlane layout, accept both **Column** and **Swimlane** constraint type.

There are two types of constraints:

1. Column
2. Swimlane

By default, the column count validation is performed based on Kanban **columns**.

Minimum card limit

The **minCount** property is used to specify the minimum cards hold on particular column or swimlane cell. If the column or swimlane total card count falls short of the minimum count value, it shows the column or cell background colour with validation fails.

Maximum card limit

The **maxCount** property is used to specify the maximum cards hold on particular column or swimlane cell. If the column or swimlane cell total card count exceeds the maximum count value, it shows the column or cell background colour with validation fails.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [

```

```

    { headerText: 'Backlog', keyField: 'Open', showItemCount: true,
minCount: 6 },
    { headerText: 'In Progress', keyField: 'InProgress', showItemCount:
true, maxCount: 5 },
    { headerText: 'Testing', keyField: 'Testing', showItemCount: true,
maxCount: 4, minCount: 3 },
    { headerText: 'Done', keyField: 'Close', showItemCount: true }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Column Validation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban column validation">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
</script>
```



```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Virtualization in EJ2 JavaScript Kanban control

Kanban allows you to load a large amount of data without any performance degradation. This feature can be enabled by setting the [enableVirtualization](#) property in the Kanban to `true`.

Virtual scrolling

Virtual scrolling optimizes data rendering within each column when using large datasets. Only a subset of cards that are visible and about to be loaded on the screen are rendered. The number of records displayed in the Kanban is determined implicitly by the height of the Kanban area and the card height. The [cardHeight](#) property of Kanban can be used to set the cards' height in pixel value. By default, the card height will be `auto`.

When the Kanban column is scrolled, the virtual scrolling feature dynamically loads additional data on demand into view and unloads the data that is no longer visible.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { generateKanbanDataVirtualScrollData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    enableVirtualization: true, // To enable virtual scrolling feature.
    dataSource: generateKanbanDataVirtualScrollData(),
    keyField: 'Status',
    enableTooltip: true,
    columns: [
        { headerText: 'To Do', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Code Review', keyField: 'Review' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        headerField: 'Id',
        contentField: 'Summary',
        selectionType: 'Multiple'
    },
    dialogSettings : {
        fields: [
            {key: 'Id', text: 'ID', type: 'TextBox'},
            {key: 'Status', text: 'Status', type: 'DropDown'},
            {key: 'StoryPoints', text: 'Story Points', type: 'Numeric' },
            {key: 'Summary', text: 'Summary', type: 'TextArea'}
        ]
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Kanban Local Data</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban Local Data">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Configure the remote data service

When the remote data is configured for the [dataSource](#), the service method will receive an additional **KanbanVirtualization** parameter to handle the initial data load for Kanban Virtualization.

To handle Kanban virtual scrolling, the server-side code needs to handle the **Where** and **Take** queries differently using the **KanbanVirtualization** parameter. The following is the example code for handling Kanban virtualization's initial data load using the **KanbanVirtualization** parameter.

```
`ts
```

```
public IActionResult LoadCard([FromBody] ExtendedDataManagerRequest dm)
{
    kanbanData = _context.KanbanCards.ToList();
    IEnumerable<KanbanCard> DataSource = kanbanData.AsEnumerable();
    DataOperations operation = new DataOperations();
    // For normal kanban data load Where query handling.
    if (dm.Where != null && dm.Where.Count > 0 && dm.KanbanVirtualization != "KanbanVirtualization")
    {
        dm.Where[0].value = dm.Where[0].value.ToString();
        DataSource = operation.PerformFiltering(DataSource, dm.Where, dm.Where[0].Operator);
    }
    if (dm.Skip != 0)
    {
        DataSource = operation.PerformSkip(DataSource, dm.Skip);
    }
    // For normal Kanban data load Take query handling.
    if (dm.Take != 0 && dm.KanbanVirtualization != "KanbanVirtualization")
    {
        DataSource = operation.PerformTake(DataSource, dm.Take);
    }
    // For Kanban virtual scrolling data load Where and Take query handling.
    var columnCount = new List<KeyValuePair<string, int>>();
    if (dm.KanbanVirtualization == "KanbanVirtualization" && dm.Where != null && dm.Where.Count > 0
        && dm.Take != 0)
    {
        IEnumerable<KanbanCard> currentData = new List<KanbanCard>();
        List<WhereFilter> currentFilter = new List<WhereFilter>();
        for (int i = 0; i < dm.Where.Count; i++)
        {
            dm.Where[i].value = dm.Where[i].value.ToString();
            currentFilter.Add(dm.Where[i]);
            var filterData = operation.PerformFiltering(DataSource, currentFilter, dm.Where[i].Operator);
            columnCount.Add(new KeyValuePair<string, int>(dm.Where[i].value.ToString(), filterData.Count()));
        }
    }
}
```

```

filterData = operation.PerformTake(filterData, dm.Take);
currentData = currentData.Concat(filterData);
currentFilter.Clear();
}
DataSource = currentData;
}
// To return the data for Kanban virtual scrolling.
if (dm.KanbanVirtualization == "KanbanVirtualization") {
return Json(new { result = DataSource, count = columnCount });
}
// To return the data for Kanban virtual scrolling.
else
{
return Json(DataSource);
}
}
`

```

Limitations for virtual scrolling

- When virtualization is enabled in a Kanban board and the card height is not explicitly set, it will not default to `auto` height. Instead, a fixed height of `100px` will be applied to the cards. It's important to note that the card height should be specified in pixel values, as percentage values are not accepted.
- When a card is dragged and dropped, the index position of the card will not be preserved when scrolling through the column.
- Virtualization is not supported for swimlanes in the Kanban board.

Localization in EJ2 JavaScript Kanban control

The localization library allows you to localize the default text content of the Kanban to different cultures using the `locale` property.

In Kanban, total count and min or max count text alone will be localized based on culture.

| Locale key | en-US (default) |

|-----|-----|

| items | items |

| min | Min |

| max | Max |

| cardsSelected | Cards Selected |

| addTitle | Add New Card |

| editTitle | Edit Card Details |

| deleteTitle | Delete Card |

| deleteContent | Are you sure you want to delete this card? |

| save | Save |

| delete | Delete |

| cancel | Cancel |

| yes | Yes |

| no | No |

| close | Close |

| noCard | No cards to display |

| unassigned | Unassigned |

Loading translations

To load translation object in an application, use `load` function of `L10n` class.

The following example demonstrates the Kanban in `Deutsch` culture.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { L10n } from '@syncfusion/ej2-base';
import { kanbanData } from './datasource.ts';
L10n.load({
  'de': {
    'kanban': {
      'items': 'Artikel',
      'min': 'Min',
      'max': 'Max',
      'cardsSelected': 'Karten ausgewählt',
      'addTitle': 'Neue Karte hinzufügen',
      'editTitle': 'Kartendetails bearbeiten',
      'deleteTitle': 'Karte löschen',
      'deleteContent': 'Möchten Sie diese Karte wirklich löschen?',
      'save': 'speichern',
      'delete': 'Löschen',
      'cancel': 'Stornieren',
      'yes': 'Ja',
      'no': 'Nein',
      'close': 'Schließen',
      'noCard': 'Keine Karten zum Anzeigen',
      'unassigned': 'nicht zugewiesen'
    }
  }
});
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
```

```

    locale: 'de',
    columns: [
        { headerText: 'Backlog', keyField: 'Open', showItemCount: true,
minCount: 6 },
        { headerText: 'In Progress', keyField: 'InProgress', showItemCount:
true, maxCount: 3 },
        { headerText: 'Done', keyField: 'Close', showItemCount: true }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    swimlaneSettings: {
        keyField: 'Assignee'
    },
    });
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Localization</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban localization">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div class="container-fluid" style="margin-top:15px;">
        <div class="row">
            <div class="col-md-9">
                <div id="Kanban"></div>
            </div>
        </div>
    </div>

```

```

        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Right to left (RTL)

The Kanban provides an option to switch its text direction and layout from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable right-to-left mode in Kanban, set the `enableRtl` to true.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { L10n, enableRtl } from '@syncfusion/ej2-base';
import { kanbanData } from './datasource.ts';
L10n.load({
    'ar': {
        'kanban': {
            'items': 'العناصر',
            'min': 'أنا',
            'max': 'ماكس',
            'cardsSelected': 'تم تحديد البطاقات',
            'addTitle': 'إضافة بطاقة جديدة',
            'editTitle': 'تحرير تفاصيل البطاقة',
            'deleteTitle': 'حذف البطاقة',
            'deleteContent': 'هل أنت متأكد أنك تريد حذف هذه البطاقة?',
            'save': 'حفظ',
            'delete': 'حذف',
            'cancel': 'إلغاء',
            'yes': 'نعم',
            'no': 'لا',
            'close': 'قريب',
            'noCard': 'لا توجد بطاقات لعرضها',
            'unassigned': 'غير معين'
        }
    }
});
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    enableRtl: true,
    locale: 'ar',
    columns: [
        { headerText: 'Backlog', keyField: 'Open', showItemCount: true,
minCount: 2 },
        { headerText: 'In Progress', keyField: 'InProgress', showItemCount:
true, maxCount: 3 },
        { headerText: 'Done', keyField: 'Close', showItemCount: true }
    ],

```

```

    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    swimlaneSettings: {
        keyField: 'Assignee'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban RTL</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban RTL mode">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div class="container-fluid" style="margin-top:15px;">
        <div class="row">
            <div class="col-md-9">
                <div id="Kanban"></div>
            </div>
        </div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```



```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Dimensions in EJ2 JavaScript Kanban control

The Kanban dimensions refers to both height and width of the entire layout and it accepts three types of values.

- Auto
- Pixel
- Percentage

Auto height and width

When height and width of the Kanban are set to **auto**, it will try as hard as possible to keep an element the same width as its parent container. In other words, the parent container that holds Kanban, its width or height will be the sum of its children. By default, Kanban is assigned with **auto** values for both the height and width properties.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  width: 'auto',
  height: 'auto',
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Auto Height and Width</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban auto height and width">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Height and width in pixel

The Kanban height and width will be rendered exactly as per the given pixel values. It accepts both string and number values.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    width: 650,
    height: '550px',
    cardSettings: {

```

```

        contentField: 'Summary',
        headerField: 'Id'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Set Kanban Height and Width on Pixel</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban height and width to Pixel">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Height and width in percentage

When height and width of the Kanban are given in percentage, it will make the Kanban as wide as the parent container.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    width: '100%',
    height: '100%',
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    }
});
kanbanObj.appendTo('#Kanban');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Set Kanban Height and Width on Percentage</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban height and width to
percentage">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Persistence in EJ2 JavaScript Kanban control

State persistence refers to the Kanban state maintained in the browser's [localStorage](#) even if the browser is refreshed or if you move to the next page within the browser.

State persistence stores Kanban datasource, column and swimlane expand/collapse state in the local storage when the [enablePersistence](#) is defined as true.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    enablePersistence: true,
    columns: [
        { headerText: 'Backlog', keyField: 'Open', allowToggle: true },
        { headerText: 'In Progress', keyField: 'InProgress', allowToggle:
true },
        { headerText: 'Testing', keyField: 'Testing', allowToggle: true },
        { headerText: 'Done', keyField: 'Close', allowToggle: true }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    swimlaneSettings: {
        keyField: 'Assignee'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Kanban Persistence</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Kanban Persistence">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Responsive mode in EJ2 JavaScript Kanban control

The Kanban component has support for responsive behavior based on the client browser's width and height.

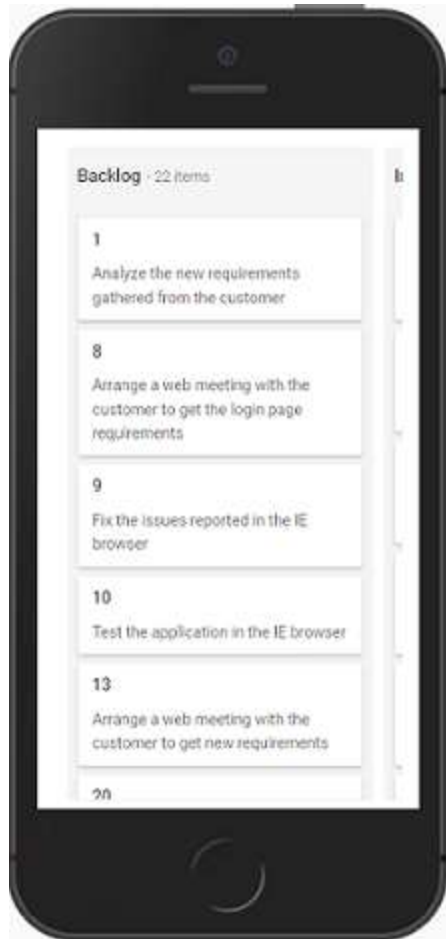
Layouts

Possible layouts are:

- Default Layout
- Swimlane Layout

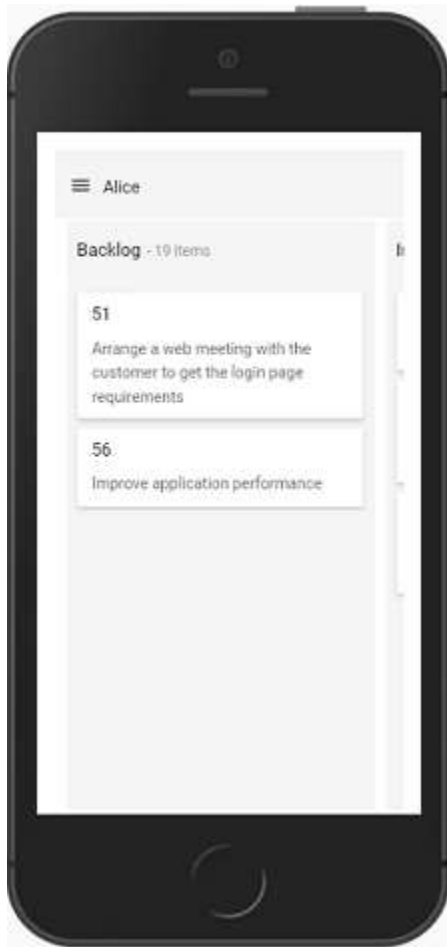
Default Layout

Kanban user interface is customized and redesigned for the best view on small screens. In responsive mode, the first column occupies 80% and the second column occupies 20% of the screen layout. Tap and hold the Kanban card to drag and drop it. Swipe left or right to view the columns.



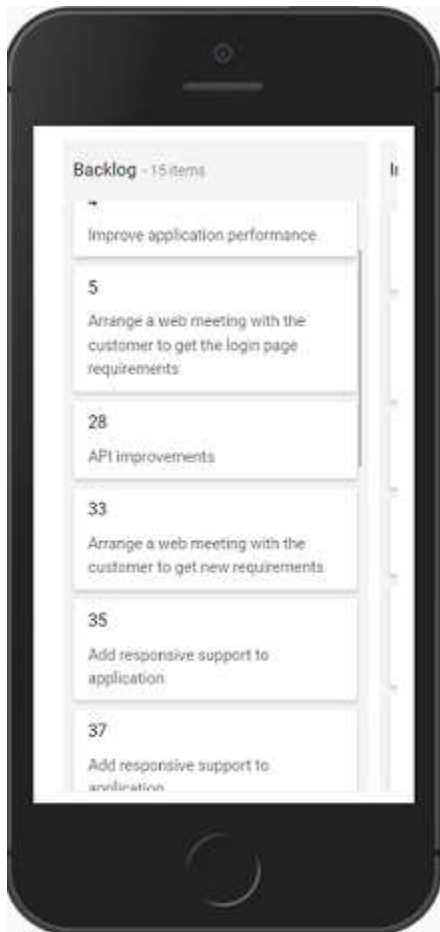
Swimlane Layout

Kanban swimlane header is rendered with menu icon on top of the kanban board. It will show all the available swimlane groups of the header text with a popup when clicking the menu icon. Swimlane selected grouped header text resultant data is rendered on the Kanban board. By default, the first swimlane grouped header text is selected and the resultant data is shown on the Kanban board. The Kanban board data will be changed when changing the swimlane group header text.



Scrolling

Column scrolling will be shown when exceeding the screen size in the columns.

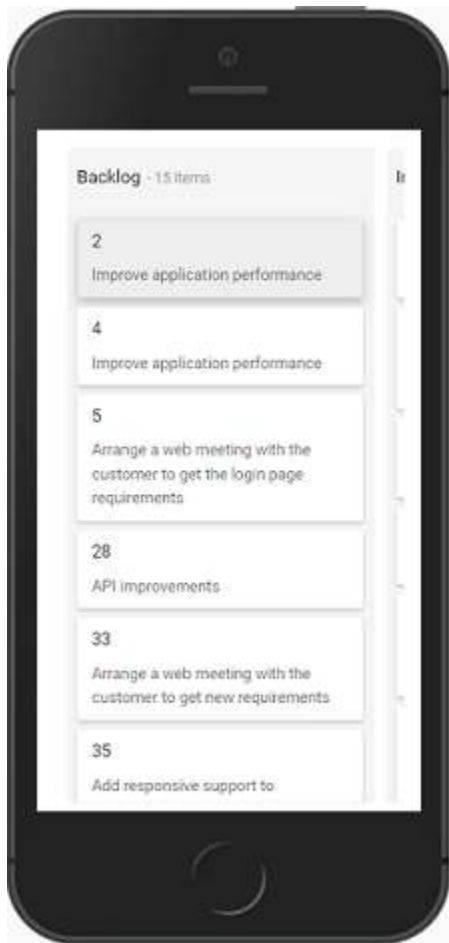


Selection

Select particular cards in the Kanban board by tapping the card.

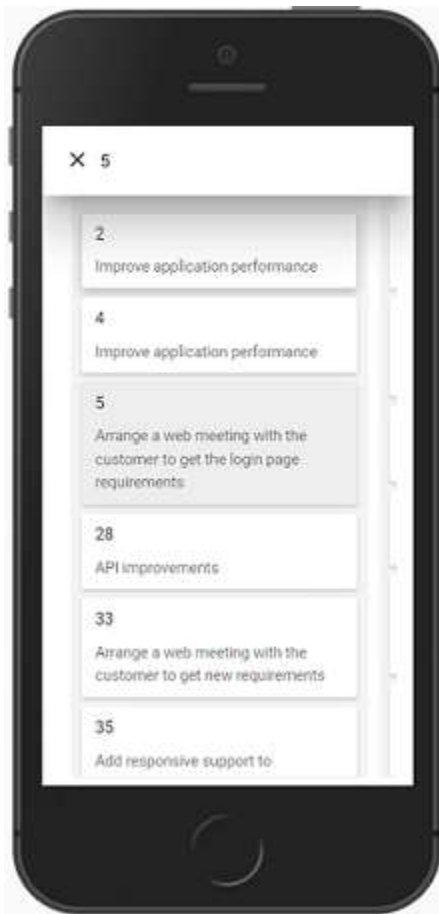
Single Selection

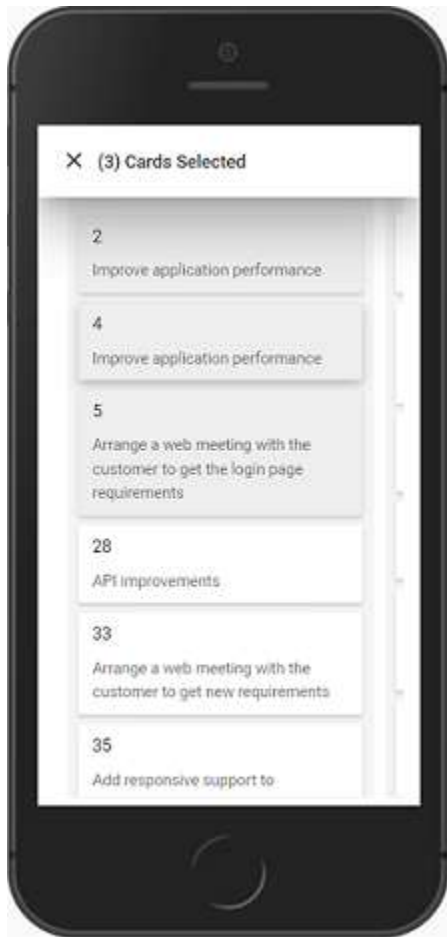
Single card will be selected when you tap the card once and selection will be removed when you select another card.



Multiple Selection

Enable [selectionType](#) as **Multiple** to select multiple cards. It will open the popup on the screen top. Selected card header text will be shown when selecting single card with a tap and hold action. If single card is selected, only tap action is required to select multiple cards. Multiple Selected card count will be shown on the popup when selecting multiple cards.





Style in EJ2 JavaScript Kanban control

To modify the Kanban appearance, you need to override the default CSS of Kanban. Also, you have an option to create your own custom theme using our [Theme Studio](#). Please find the list of CSS classes in Kanban.

Css class	Purpose
-----	-----
.e-kanban .e-kanban-table	customize the kanban.
.e-kanban .e-kanban-header .e-header-cells	Header cells of kanban.
.e-kanban .e-kanban-header .e-header-cells .e-header-wrap .e-header-title	Header title of kanban.
.e-kanban .e-kanban-header .e-header-cells.e-min-color	Header cells minimum color of kanban.
.e-kanban .e-kanban-header .e-header-cells.e-max-color	Header cells maximum color of kanban.
.e-kanban .e-kanban-header .e-header-cells.e-collapsed.e-min-color	Header cells of collapsed column minimum color in column constraint type of kanban.
.e-kanban .e-kanban-header .e-header-cells.e-collapsed.e-max-color	Header cells of collapsed column maximum color in column constraint type of kanban.
.e-kanban .e-kanban-header .e-header-cells .e-header-text	Header text of Kanban.

| .e-kanban .e-kanban-header .e-header-cells .e-item-count | Header cells Item count of Kanban. |

| .e-kanban .e-kanban-header .e-header-cells .e-limits | Header cells limits in column constraint type of kanban. |

| .e-kanban .e-kanban-header .e-header-cells .e-limits .e-min-count | Header cells minimum count of kanban. |

| .e-kanban .e-kanban-header .e-header-cells .e-limits .e-max-count | Header cells maximum count of kanban. |

| .e-kanban .e-kanban-content | Customize kanban Content. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-limits | Content cells limits in swimlane constraint type of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-limits .e-min-count | Content cells minimum count of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-limits .e-max-count | Content cells maximum count of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-min-color | Content cells minimum color of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-max-color | Content cells maximum color of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-collapsed .e-collapse-header-text | Content cells of collapsed header text. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-collapsed .e-collapse-header-text .e-item-count | Content cells of collapsed header text Item count. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-show-add-button | Add button in content cells of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-show-add-button .e-show-add-icon | Customize content cells add icon of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-empty-card | Empty content cells of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card | Customize cards in kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-header .e-card-header-title | Cards header title of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-footer | Cards footer of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-content | Cards content of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card.e-card-color | Cards color of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-tags |
Customize Card tags of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-tag |
Card tag of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-swimlane-row-expand | Content cells of swimlane row expand of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-swimlane-row-collapse | Content cells of swimlane row collapse of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-swimlane-text | Content cells of swimlane header text of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-item-count | Content cells of swimlane items count of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells | swimlane content cells of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells.e-dropping |
Customize swimlane content cells card dropping of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells .e-card-wrapper |
Swimlane content cells of card wrapper. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells.e-min-color |
Swimlane content cells of minimum color of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells.e-max-color |
Swimlane content cells of maximum color of kanban. |Customize the kanban CSS theme. Please find the list of CSS classes in Kanban. |

| .e-kanban .e-kanban-table .e-header-cells .e-header-text | Header text of Kanban. |

| .e-kanban .e-kanban-table .e-header-cells .e-item-count | Header cells Item count of Kanban. |

| .e-kanban .e-kanban-table .e-header-cells .e-column-expand | Header cells of toggle icon in column expand. |

| .e-kanban .e-kanban-table .e-header-cells .e-column-collapse | Header cells of toggle icon in column collapse. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row:not(.e-swimlane-row) .e-content-cells |
swimlane content cells of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row.e-swimlane-row .e-swimlane-text | Content cells of swimlane header text of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row.e-swimlane-row .e-item-count | Content cells of swimlane items count of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row .e-show-add-button .e-show-add-icon | Add icon in content cells of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-card.e-selection | Selected card of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-card .e-card-header | Cards header in kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-card .e-card-content | Cards content in kanban. |
 | .e-kanban .e-kanban-table.e-content-table .e-card .e-card-tag.e-card-label | Cards label in kanban. |

To set fixed position to the Kanban header

The Fixed header in Kanban control can be customized in following ways,

By setting a fixed height to the Kanban content,

```
`css
.e-kanban .e-kanban-content {
height: 500px;
}
`
```

By customizing the CSS for the Kanban header.

```
`css
.e-kanban-header {
position: -webkit-sticky;
position: sticky;
z-index: 100;
top: 0;
}
`
```

Note: It will not affect the Kanban content's height.

Accessibility in EJ2 JavaScript Kanban control

The Kanban component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties. This component is characterized by complete ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The accessibility compliance for the Kanban component is outlined below.

Accessibility Criteria	Compatibility
--	--
WCAG 2.2 Support	![Yes](https://cdn.syncfusion.com/content/images/landing-page/yes.png)
Section 508 Support	![Yes](https://cdn.syncfusion.com/content/images/landing-page/yes.png)
Screen Reader Support	![Yes](https://cdn.syncfusion.com/content/images/landing-page/yes.png)
Right-To-Left Support	![Yes](https://cdn.syncfusion.com/content/images/landing-page/yes.png)
Color Contrast	![Intermediate](https://cdn.syncfusion.com/content/images/landing-page/intermediate.png)
Mobile Device Support	![Yes](https://cdn.syncfusion.com/content/images/landing-page/yes.png)

| [Keyboard Navigation Support](#) | ![Yes](https://cdn.syncfusion.com/content/images/landing-page/yes.png) |

| [Accessibility Checker Validation](#) |
 ![Intermediate](https://cdn.syncfusion.com/content/images/landing-page/intermediate.png) |

| [Axe-core Accessibility Validation](#) | ![Yes](https://cdn.syncfusion.com/content/images/landing-page/yes.png) |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

![Yes](https://cdn.syncfusion.com/content/images/landing-page/yes.png) - All features of the component meet the requirement.

![Intermediate](https://cdn.syncfusion.com/content/images/landing-page/intermediate.png) - Some features of the component do not meet the requirement.

![No](https://cdn.syncfusion.com/content/images/landing-page/no.png) - The component does not meet the requirement.

WAI-ARIA attributes

The Kanban component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Kanban component:

| Attributes | Purpose |

| --- | --- |

| **aria-label** | It helps to provides information about elements in a kanban component for assistive technology. |

| **aria-expanded** | Attributes indicate the state of a collapsible element. |

| **aria-selected** | This attribute is assigned to the Kanban component for the selection of elements, and its default value is **false**. The value changes to true when the user selects a Kanban card. |

| **aria-grabbed** | Indicates whether the attribute is set to true. It has been selected for dragging. If this attribute is set to false, the element can be grabbed for a drag-and-drop operation but will not be currently grabbed. |

| **aria-describedby** | This attribute contains the ID of the Kanban header column to indicate that the attribute establishes an association between the Kanban header column and the Kanban column body. |

| **aria-roledescription** | This attribute is assigned to the Kanban component and is used to provide alternative descriptions for card elements. |

Keyboard interaction

The Kanban component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Kanban component.

| **Press** | **To do this** |

| --- | --- |

| **Home** | **To select the first card in the kanban** |

| **End** | **To select the last card in the kanban** |

| **Arrow Up** | **Select the card through the up arrow** |

| **Arrow Down** | **Select the card through the down arrow** |

| **Arrow Right** | **Move the column selection to the right** |

| **Arrow Left** | **Move the column selection to the left** |

| **Ctrl + Enter** | **Used to select the multi cards** |

| **Ctrl + Space** | **Used to select the multi cards** |

| **Shift + Arrow Up** | **Used to select the multiple cards towards up** |

| **Shift + Arrow Down** | **Used to select the multiple cards towards down** |

| **Shift + Tab** | **Reverse order of the tab action** |

| **Enter** | **Open the selected cards** |

| **Tab** | **To navigate the Kanban column** |

| **Delete** | **To delete the selected cards** |

| **ESC** | **Escape from the modified details** |

| **Space** | **Used to open the card edit dialog based on the column selection** |

Disable keyboard interaction

Disables all the functionalities in the Kanban board performed using keyboard by setting the `allowKeyboard` properties to `false`.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  allowKeyboard: false,
  columns: [
    { headerText: 'Backlog', keyField: 'Open', allowToggle: true },
    { headerText: 'In Progress', keyField: 'InProgress', allowToggle:
true },
    { headerText: 'Testing', keyField: 'Testing', allowToggle: true },
    { headerText: 'Done', keyField: 'Close', allowToggle: true }
  ],
});
```

```

    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    swimlaneSettings: {
        keyField: 'Assignee'
    }
});
kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Disable Keyboard Functionalities</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Disable Kanban keyboard
functionalities">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Kanban"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Ensuring accessibility

The Kanban component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Kanban component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Kanban component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

How To

Header double click in EJ2 JavaScript Kanban control

You can bind the header double click event by using the [dataBound](#) event at the initial rendering. You can get the column header text when you double click on the headers.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { DialogUtility } from '@syncfusion/ej2-popups';
import { closest } from '@syncfusion/ej2-base';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    },
    dataBound: function () {
        let headerEle: HTMLElement = document.querySelector('.e-header-row');
        headerEle.addEventListener("dblclick", function (e: Event) {
            let target = closest((<HTMLElement>e.target), '.e-header-cells');
            DialogUtility.alert({
                title: 'Header',
                content: "Double clicked on " +
                    (<HTMLElement>target.querySelector('.e-header-text')).innerText + " header",
                showCloseIcon: true,
                closeOnEscape: true,
                animationSettings: { effect: 'Zoom' }
            });
        });
    }
});
```

```

    }
  });
  kanbanObj.appendTo('#Kanban');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Auto Height and Width</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Kanban auto height and width">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="content-wrapper">
      <div id="Kanban"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamically change columns in EJ2 JavaScript Kanban control

You can dynamically change the Kanban columns by using the [columns](#) property.

In the below sample, you can dynamically change the [allowToggle](#) property at the particular column when you click on the button. You can also change the initially created columns to the new Kanban columns by using the [columns](#) property when you click on the button.

INDEX.TS

```
import { Kanban } from '@syncfusion/ej2-kanban';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {
        contentField: 'Summary',
        headerField: 'Id'
    }
});
kanbanObj.appendTo('#Kanban');
let particularColumn: HTMLElement =
document.getElementById('particularColumn');
let column: HTMLElement = document.getElementById('column');
particularColumn.onclick = () => {
    kanbanObj.columns[1].allowToggle= true;
};
column.onclick = () => {
    kanbanObj.columns = [
        { headerText: 'To Do', keyField: 'Open' },
        { headerText: 'Done', keyField: 'Close' }
    ]
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>Kanban Card without Header</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Kanban card without header">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <button class="e-btn" id="particularColumn">Enable Allow
Toggle</button>
            <button class="e-btn" id="column">Change Columns</button>
            <div id="Kanban"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter cards in EJ2 JavaScript Kanban control

You can filter the collection of cards from the dataSource and display it on the Kanban board by using the [query](#) property.

In the below sample, you can filter the cards based on the 'where' query and display the filtered data to the Kanban board.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
import { Query } from '@syncfusion/ej2-data';
import { kanbanData } from './datasource.ts';
let kanbanObj: Kanban = new Kanban({
    dataSource: kanbanData,
    keyField: 'Status',
    columns: [
        { headerText: 'Backlog', keyField: 'Open' },
        { headerText: 'In Progress', keyField: 'InProgress' },
        { headerText: 'Testing', keyField: 'Testing' },
        { headerText: 'Done', keyField: 'Close' }
    ],
    cardSettings: {

```

```

        contentField: 'Summary',
        headerField: 'Id'
    }
});
kanbanObj.appendTo('#Kanban');
let priorityObj: DropDownList = new DropDownList({
    dataSource: ['None', 'High', 'Normal', 'Low'],
    index: 0,
    placeholder: 'Select a priority',
    width: 100,
    change: change
});
priorityObj.appendTo('#filter');
function change(args: ChangeEventArgs): void {
    let filterQuery: Query = new Query();
    if (args.value !== 'None') {
        filterQuery = new Query().where('Priority', 'equal', args.value);
    }
    (kanbanObj as any).query = filterQuery;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Card without Header</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban card without header">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div class="content-wrapper">
    <input id="filter" type="text">
      <div id="Kanban"></div>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Search cards in EJ2 JavaScript Kanban control

You can search the cards in Kanban by using the `query` property.

In the following sample, the searching operation starts as soon as you start typing characters in the external text box. It will search the cards based on the `Id` and `Summary` using the `search` query with `contains` operator.

INDEX.TS

```

import { Kanban } from '@syncfusion/ej2-kanban';
import { TextBox } from '@syncfusion/ej2-inputs';
import { Query } from '@syncfusion/ej2-data';
import { kanbanData } from '../datasource.ts';
let kanbanObj: Kanban = new Kanban({
  dataSource: kanbanData,
  keyField: 'Status',
  columns: [
    { headerText: 'Backlog', keyField: 'Open' },
    { headerText: 'In Progress', keyField: 'InProgress' },
    { headerText: 'Testing', keyField: 'Testing' },
    { headerText: 'Done', keyField: 'Close' }
  ],
  cardSettings: {
    contentField: 'Summary',
    headerField: 'Id'
  }
});
kanbanObj.appendTo('#Kanban');
let textObj: TextBox = new TextBox({
  placeholder: 'Enter search text',
  showClearButton: true,
  width: 180
});
textObj.appendTo('#search');
document.getElementById('reset').onclick = () => {
  textObj.value = '';
  kanbanObj.query = new Query();
};
document.getElementById('search').onkeyup = (e: KeyboardEvent) => {
  let searchValue: string = (<HTMLInputElement>e.target).value;
  let searchQuery: Query = new Query();

```



```

    if (searchValue !== '') {
        searchQuery = new Query().search(searchValue, ['Id', 'Summary'],
        'contains', true);
    }
    kanbanObj.query = searchQuery;
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Kanban Card without Header</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Kanban card without header">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
kanban/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <table>
                <tbody>
                    <tr><td style="width: 200px">
                        <input id="search" class="e-input"
placeholder="Enter search text" required="">
                    </td>
                    <td><button class="e-btn" id="reset">Reset</button></td>
                </tr></tbody>
            </table>
            <div id="Kanban"></div>
        </div>
    </div>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Ej1 api migration in EJ2 JavaScript Kanban control

This article describes the API migration process of Kanban component from Essential JS 1 to Essential JS 2.

Columns

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **Default** | **Property** : *columns* </br></br> \$("#Kanban").ejKanban({ </br> columns : [] }); | **Property** : *columns* </br></br> var obj = new </br>ej.kanban.Kanban({ </br>columns : [] }) </br>obj.appendTo("#Kanban"); |

| **Header Text** | **Property** : *headerText* </br></br> \$("#Kanban").ejKanban({ </br>columns: [{ </br>headerText : "Backlog"}] </br> }); | **Property** : *headerText* </br> </br> var obj = new </br>ej.kanban.Kanban({ columns: [{ </br>headerText : "Backlog"}] </br> }); </br>obj.appendTo("#Kanban"); |

| **Key Field** | **Property** : *key* </br></br> \$("#Kanban").ejKanban({ columns: [{ </br>key : "Open"}] }); | **Property** : *keyField* </br></br> var obj = new </br>ej.kanban.Kanban({ columns: [{ </br>keyField : "Open"}] }); </br>obj.appendTo("#Kanban"); |

| **Initial Collapsed Columns** | **Property** : *isCollapsed* </br></br> \$("#Kanban").ejKanban({ </br>columns: [{ </br>headerText: "Backlog", </br>key: "Open", </br>isCollapsed : true}] }); | **Property** : *isExpanded* </br></br> var obj= new </br>ej.kanban.Kanban({ columns:[{ </br>headerText: "Backlog", </br>keyField: "Open", </br>isExpanded : true}] }); </br>obj.appendTo("#kanban"); |

| **Cell Add card button** | **Property** : *showAddButton* </br></br> \$("#Kanban").ejKanban({ </br>columns: [{ </br>headerText: "Backlog", </br>key: "Open", </br>showAddButton : true}] }); | **Property** : *showAddButton* </br> </br> var obj = new </br>ej.kanban.Kanban({ columns: [{ </br>headerText: "Backlog", </br>keyField: "Open", </br>showAddButton : true}] }); </br>obj.appendTo("#kanban"); |

| **Column card count** | **Property** : *enableTotalCount* </br></br> \$("#Kanban").ejKanban({ </br>enableTotalCount : true}); | **Property** : *showItemCount* </br> </br> var obj = new </br>ej.kanban.Kanban({ columns: [{ </br>headerText: "Backlog", </br>keyField: "Open", </br>showItemCount : true}] }); </br>obj.appendTo("#kanban"); |

| **Template** | **Property** : *headerTemplate* </br></br> \$("#Kanban").ejKanban({ </br>columns: [{ </br>headerText: "Backlog", </br>key: "Open", </br>headerTemplate : "#template"}] }); | **Property** : *template* </br></br> var obj = new </br>ej.kanban.Kanban({ columns: [{ </br>template : "#headerTemplate"}] }); </br>obj.appendTo("#kanban"); |

| **Allow Drop** | **Property** : *allowDrop* </br> </br> \$("#Kanban").ejKanban({ </br>columns: [{ </br>headerText: "Backlog", </br>key: "Open", </br>allowDrop: false}] }); | **Not Available** |

| Allow Drag | **Property:** *allowDrag* </br> </br>\$("#Kanban").ejKanban({</br>columns: [{</br>headerText: "Backlog",</br> key: "Open",</br>allowDrag : false</br>}]); | **Not Available** |

| Total Count text | **Property:** *totalCount* </br> </br>\$("#Kanban").ejKanban({</br>enableTotalCount: true,</br> columns: [{</br> headerText: "Backlog",</br> key: "Open", </br>totalCount : </br>{text: "Backlog Count"}</br>}]); | **Not Available** |

| Width | **Property:** *width*</br></br>\$("#Kanban").ejKanban({</br>enableTotalCount: true,</br> columns: [{</br> headerText: "Backlog",</br> key: "Open", </br>width : 200</br>}]); | **Not Available** |

| Visible | **Property:** *visible*</br></br>\$("#Kanban").ejKanban({</br>enableTotalCount: true,</br>columns: [{</br>headerText: "Backlog",</br> key: "Open", </br>visible : false</br>}]); | **Not Available** |

| Add/Delete Columns | **Method:** *columns(column, key,</br> [action])*</br></br>**Add :**</br>var kanbanObj = </br>\$("#Kanban").data("ejKanban");</br>kanbanObj.columns</br>("Review", "Review", "add");</br></br>**Delete:** </br>var kanbanObj = </br>\$("#Kanban").data("ejKanban");</br>kanbanObj.columns</br>("Review", "Review", "remove");</br> | **Method:** *addColumn(columnOptions,</br> index)*</br></br>var obj = new </br>ej.kanban.Kanban({});</br>obj.appendTo("#kanban");</br>obj.addColumn({</br>headerText: "Review",</br> keyField: "Review"</br>}, 2);</br></br>**Method:** *deleteColumn(index)* </br></br>var obj = new </br>ej.kanban.Kanban({});</br>obj.appendTo("#kanban");</br>obj.deleteColumn(2); |

| Show Columns | **Method:** *showColumns(headerText)* </br></br>var kanbanObj = </br>\$("#Kanban")</br>.data("ejKanban");</br>kanbanObj.</br>showColumns("Testing"); | **Method:** *showColumn(key)* </br></br>var obj = </br>new ej.kanban.Kanban({});</br>obj.appendTo("#kanban");</br>obj.showColumn</br>("Testing"); |

| Hide Column | **Method:** *hideColumns(headerText)* </br></br>var kanbanObj = </br>\$("#Kanban")</br>.data("ejKanban");</br>kanbanObj.</br>hideColumns("Testing"); | **Method:** *hideColumn(key)*</br></br>var obj = </br>new ej.kanban.Kanban({});</br>obj.appendTo("#kanban");</br>obj.hideColumn("Testing"); |

| Get Visible Column Names | **Method:** *getVisibleColumnNames()*</br></br>var kanbanObj = </br>\$("#Kanban")</br>.data("ejKanban");</br>kanbanObj.</br>getVisibleColumnNames(); | **Not Applicable** |

| Get Column By</br>Header Text | **Method:** *getColumnByHeaderText</br>(headerText)*</br></br>var kanbanObj = </br>\$("#Kanban")</br>.data("ejKanban");</br>kanbanObj.</br>getColumnByHeaderText</br>("Testing"); | **Not Applicable** |

| Get Column Data | **Not Applicable** | **Method:** *getColumnData()*</br></br>var obj = </br>new ej.kanban.Kanban({});</br>obj.appendTo("#kanban");</br>obj.getColumnData(); |

| Triggers after</br>cell is click | **Event:** *cellClick*</br></br>\$("#Kanban").ejKanban({</br>cellClick : function (args) {}}); | **Not Applicable** |

Cards

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Card unique field | **Property :** </br>*fields.primaryKey*</br></br>\$("#Kanban").ejKanban({</br> fields: { primaryKey: "Id"</br> } }); | **Property :** </br>*cardSettings.headerField*</br>var obj = </br>new

```

ej.kanban.Kanban({ cardSettings: { headerField: "Id" } });obj.appendTo("#kanban");
|
| Content | Property: fields.content $("#Kanban").ejKanban({ fields: {
content: "Summary" } }; | Property: cardSettings.contentField var obj =
new ej.kanban.Kanban({ cardSettings: { contentField: "Id" }
});obj.appendTo("#kanban") |
| Tag | Property: fields.tag $("#Kanban").ejKanban({ fields: { tag:
"Tags" } }; | Property: cardSettings.tagsField var obj = new ej.kanban.Kanban({
cardSettings: { tagsField: "Tags" } }; obj.appendTo("#kanban"); |
| Left border color | Property: fields.color $("#Kanban").ejKanban({ fields: {
color: "Type" }, cardSettings: { colorMapping: { "cb2027": "Issue,Story",
"#67ab47": "Improvement" } }; | Property: cardSettings.grabberField var obj = new
ej.kanban.Kanban({ cardSettings: { grabberField: "color" }
});obj.appendTo("#kanban"); |
| Header | Property: fields.title $("#Kanban").ejKanban({ fields: { title:
"Assignee" } }; | Card Unique mapping field is displayed on card header. |
| Image | Property: fields.imageUrl $("#Kanban").ejKanban({ fields: {
imageUrl: "ImgUrl" } }; | Not Applicable |
| CSS class | Not Applicable | Property : cardSettings.footerCssField var obj = new
ej.kanban.Kanban({ cardSettings: { footerCssField: "classNames" }
});obj.appendTo("#kanban"); |
| Template | Property: cardSettings.template $("#Kanban").ejKanban({ cardSettings:
{ template: "#cardTemplate" } }; | Property: cardSettings.template var obj =
new ej.kanban.Kanban({ cardSettings: { template:
"#cardTemplate" } }; obj.appendTo("#kanban"); |
| Toggle Card | Method: toggleCard($div or id) var kanbanObj =
$("#Kanban").data("ejKanban");kanbanObj.toggleCard("2"); | Not Applicable |
| Get Card Details | Not Applicable | Method: getCardDetails(target) var obj = new
ej.kanban.Kanban({});obj.appendTo("#kanban");obj.getCardDetails(obj.element
querySelector(".e-card")); |
| Get Selected Cards | Not Applicable | Method: getSelectedCards() var obj = new
ej.kanban.Kanban({});obj.appendTo("#kanban");obj.getSelectedCards(); |
| Card Click | Event: cardClick $("#Kanban").ejKanban({ cardClick: function (args)
{ } }; | Event: cardClick var obj = new ej.kanban.Kanban({ cardClick:
function(args) { } }; |
| Card Double Click | Event:
cardDoubleClick $("#Kanban").ejKanban({ cardDoubleClick: function (args) { } }; |
Event: cardDoubleClick var obj = new
ej.kanban.Kanban({ cardDoubleClick: function(args) { } }; |
| Triggers when start the drag | Event: cardDragStart $("#Kanban").ejKanban({
cardDragStart: function (args) { } }; | Event: dragStart var obj = new ej.kanban.Kanban({
dragStart: function(args) { } }; |

```

| Triggers when card is dragged | **Event:**

cardDrag | **Event:** *drag*
 var obj = new ej.kanban.Kanban({ drag: function(args) {} }); |

| Triggers when card dragging stops | **Event:**

cardDragStop | **Event:** *dragStop*
 var obj = new ej.kanban.Kanban({ dragStop: function(args) {} }); |

| Triggers after save the data when dropped | **Event:**

cardDrop | **Not Applicable**
 |

| Triggers after cell is click | **Event:** *cellClick* | **Not Applicable** |

| Triggers each card rendered | **Event:**

queryCellInfo | **Event:** *cardRendered*
 var obj = new ej.kanban.Kanban({ cardRendered: function(args) {} }); |

DataSource

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **DataSource** | **Property:** *dataSource* | **Method:** *dataSource(datasource)*
 var kanbanObj = \$("#kanban").ejKanban({ dataSource: new DataSource() }); | **Property:** *dataSource*
 var obj = new ej.kanban.Kanban({ datasource: new DataSource() }); | **Method:** *dataSource(datasource)*
 var obj = new ej.kanban.Kanban({}); obj.appendTo("#kanban"); obj.dataSource(new DataSource()); |

| Triggers before data load | **Event:** *load* | **Event:** *dataBinding*
 var obj = new ej.kanban.Kanban({ load: function(args) {} }); | **Event:** *dataBinding*
 var obj = new ej.kanban.Kanban({ dataBinding: function(args) {} }); |

| Triggers after data bounded | **Event:**

dataBound | **Event:** *dataBound*
 var obj = new ej.kanban.Kanban({ dataBound: function(args) {} }); |

Common:

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Drag And Drop | **Property:**

allowDragAndDrop | **Property:** *allowDragAndDrop*
 var obj = new ej.kanban.Kanban({ allowDragAndDrop: true }); | **Property:** *allowDragAndDrop*
 var obj = new ej.kanban.Kanban({ allowDragAndDrop: true }); obj.appendTo("#kanban"); |

| Key Field | **Property:** *keyField* | **Property:** *keyField*
 var obj = new ej.kanban.Kanban({ keyField: "Status" }); | **Property:** *keyField*
 var obj = new ej.kanban.Kanban({ keyField: "Status" }); obj.appendTo("#kanban"); |

| Title | **Property:** *allowTitle*
 | **Applicable** |
 | CssClass | **Property:** *cssClass*
 | **Property:** *cssClass*
 var obj = new ej.kanban.Kanban({cssClass : "custom-class"});
 obj.appendTo("#kanban"); |
 | Print | **Property:** *allowPrinting*
 true});
 | **Method:** *print()*
 var kanbanObj = \$("#Kanban").data("ejKanban");
 kanbanObj.print(); | **Not Applicable** |
 | Touch | **Property:** *enableTouch*
 | **Applicable** |
 | Locale | **Property:** *locale*
 | **Property:** *locale*
 var obj = new ej.kanban.Kanban({locale: "de-DE"});
 obj.appendTo("#kanban"); |
 | Query | **Property:** *query*
 ej.Query().select(["Status", "Id", "Summary"]); | **Property :** *query*
 var obj = new ej.kanban.Kanban({query: new Query().select("")});
 obj.appendTo("#kanban"); |
 | Refresh | **Method:** *refresh([templateRefresh])*
 var kanbanObj = \$("#Kanban").data("ejKanban");
 kanbanObj.refresh(); | **Method:** *refresh()*
 var obj = new ej.kanban.Kanban({});
 obj.appendTo("#kanban");
 obj.refresh(); |
 | Refresh Template | **Method:** *refreshTemplate()*
 var kanbanObj = \$("#Kanban").data("ejKanban");
 kanbanObj.refreshTemplate(); | **Not Applicable** |
 | Destroy | **Method:** *destroy()*
 var kanbanObj = \$("#Kanban").data("ejKanban");
 kanbanObj.destroy(); | **Method:** *destroy()*
 var obj = new ej.kanban.Kanban({});
 obj.appendTo("#kanban");
 obj.destroy(); |
 | Get Header Table | **Method:** *getHeaderTable()*
 var kanbanObj = \$("#Kanban").data("ejKanban");
 kanbanObj.getHeaderTable(); | **Not Applicable** |
 | Show Spinner | **Not Applicable** | **Method:** *showSpinner()*
 var obj = new ej.kanban.Kanban({});
 obj.appendTo("#kanban");
 obj.showSpinner(); |
 | Hide Spinner | **Not Applicable** | **Method:** *hideSpinner()*
 var obj = new ej.kanban.Kanban({});
 obj.appendTo("#kanban");
 obj.hideSpinner(); |
 | Triggers before every action | **Event:** *actionBegin*
 | **Event:** *actionBegin*
 var obj = new ej.kanban.Kanban({actionBegin: function(args) {} }); |
 | Triggers on successfully completion of actions | **Event:** *actionComplete*
 | **Event:** *actionComplete*
 var obj = new ej.kanban.Kanban({actionComplete: function(args) {} }); |
 | Triggers on action failure | **Event:** *actionFailure*
 | **Event:** *actionFailure*
 var obj = new ej.kanban.Kanban({actionFailure: function(args) {} }); |

| Triggers after
Kanban rendered | **Event:** *create*

\$("#Kanban").ejKanban({
create: function (args) {} }); | **Event:** *created*

var obj = new ej.kanban.Kanban({
created: function(args){} }); |

| Triggers when
header click | **Event:**

headerClick

\$("#Kanban").ejKanban({
headerClick: function (args) {} }); | **Not Applicable** |

| Triggers when
destroy | **Event:** *destroy*

\$("#Kanban").ejKanban({
destroy: function (args) {} }); | **Event:** *destroy*

var obj = new ej.kanban.Kanban({
destroy: function(args){} }); |

Swimlane:

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** *swimlaneKey*

\$("#Kanban").ejKanban({
fields: {
swimlaneKey: "Assignee"} }); | **Property:** *keyField*

var obj = new ej.kanban.Kanban({
swimlaneSettings: {
keyField: "Assignee" } });
obj.appendTo("#kanban"); |

| Header | **Property:** *headers*

\$("#Kanban").ejKanban({
headers: [{
text: "Andrew",
key: "Andrew Fuller" }] }); | **Property:** *textField*

var obj = new ej.kanban.Kanban({
swimlaneSettings: {
textField: "AssigneeName" } });
obj.appendTo("#kanban"); |

| Drag And Drop | **Property:** *allowDragAndDrop*

\$("#Kanban").ejKanban({
swimlaneSettings: {
allowDragAndDrop: true } }); | **Property:** *allowDragAndDrop*

var obj = new ej.kanban.Kanban({
swimlaneSettings: {
allowDragAndDrop: true } });
obj.appendTo("#kanban"); |

| Card Count | **Property:** *showCount*

\$("#Kanban").ejKanban({
swimlaneSettings: {
showCount: true } }); | **Property:** *showItemCount*

var obj = new ej.kanban.Kanban({
swimlaneSettings: {
showItemCount: true } });
obj.appendTo("#kanban"); |

| Empty Row | **Property:**
showEmptySwimlane

\$("#Kanban").ejKanban({
swimlaneSettings: {
showEmptySwimlane: true } }); | **Property:** *showEmptyRow*

var obj = new ej.kanban.Kanban({
swimlaneSettings: {
showEmptyRow: true } });
obj.appendTo("#kanban"); |

| Sorting | **Not Available** | **Property:**
sortDirection

var obj = new
ej.kanban.Kanban({
swimlaneSettings: {
sortDirection:
"Descending" } });
obj.appendTo("#kanban"); |

| Expand All | **Method:** *expandAll()*

var kanbanObj = \$("#Kanban")
.data("ejKanban");
kanbanObj.KanbanSwimlane
.expandAll(); | **Not Applicable** |

| Collapse All | **Method:** *collapseAll()*

var kanbanObj = \$("#Kanban")
.data("ejKanban");
kanbanObj.KanbanSwimlane
.collapseAll(); | **Not Applicable** |

| Toggle | **Method:** *toggle(\$div or key)*

var kanbanObj = \$("#Kanban")
.data("ejKanban");
kanbanObj.KanbanSwimlane
.toggle(\$(".e-slexpandcollapse")
.eq(1)); | **Not Applicable** |

| Get Swimlane Data | **Not Applicable** | **Method:** `</br>getSwimlaneData(keyField)</br></br>var obj = new ej.kanban.Kanban({});</br>obj.appendTo("#kanban");</br>obj.</br>getSwimlaneData("Janet"); |`

| Triggers before swimlane</br>icon click event | **Event:** `swimlaneClick</br></br>$("#Kanban").ejKanban({</br>swimlaneClick:</br>function (args) {}}); |` **Not Applicable** |

Stacked Headers

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Multiple stacked headers | **Property:** `stackedHeaderColumns</br></br>$("#Kanban").ejKanban({</br>stackedHeaderRows: [{</br>stackedHeaderColumns: [{</br>headerText: "Status",</br>column: "Backlog,</br>In Progress, Testing,</br>Done"}] },</br> { stackedHeaderColumns: [{</br>headerText: "Unresolved",</br>column: "Backlog,</br>In Progress"}]}}}); |` **Not Applicable** |

| Single Stacked Header | **Property:** `stackedHeaderColumns</br></br>$("#Kanban").ejKanban({</br>stackedHeaderRows: [{</br>stackedHeaderColumns: [{</br>headerText: "Unresolved",</br>column: "Backlog,</br>In Progress"}]}}}); |` **Property:** `</br>stackedHeaders</br>var obj = new </br>ej.kanban.Kanban({</br>stackedHeaders: [{</br>text: "To Do",</br>keyField: "Open,</br>InProgress"}]};</br>obj.appendTo("#kanban"); |`

WIP Validation

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Constraints Type | **Property:** `</br>constraints.type</br></br>$("#Kanban").ejKanban({</br>columns: [{</br>headerText: "Backlog",</br>key: "Open",</br>constraints: {</br>type: "swimlane", max: 5}}]); |` **Property:** `</br>constraintType</br></br>var kanban = new</br>ej.kanban.Kanban({</br>constraintType:</br>"swimlane" }); |`

| Maximum card Count at</br>particular column/swimlane | **Property:** `</br>constraints.max</br></br>$("#Kanban").ejKanban({</br>columns: [{</br>headerText: "Backlog",</br>key: "Open",</br>constraints: {</br>type: "swimlane",</br>max: 5}}]); |` **Property:** `</br>maxCount</br></br>var obj = new ej.kanban.Kanban({</br>columns:[</br>headerText: "Backlog",</br>keyField: "Open",</br>maxCount: 5}]};</br>obj.appendTo("#kanban"); |`

| Minimum card Count at</br>particular column | **Property:** `</br>constraints.min</br></br>$("#Kanban").ejKanban({</br>columns: [{</br>headerText: "Backlog",</br>key: "Open",</br>constraints: {</br>type: "swimlane",</br>min: 2} }]); |` **Property:** `minCount</br>var obj = new </br>ej.kanban.Kanban({</br>columns:[</br>headerText: "Backlog",</br>keyField: "Open",</br>minCount: 2}]};</br>obj.appendTo("#kanban"); |`

Keyboard

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| KeyBoard | **Property:** `</br>allowKeyboardNavigation</br></br>$("#Kanban").ejKanban({</br>allowKeyboardNavigation:</br>true}); |` **Property:** `</br>allowKeyboard</br></br>var obj = new </br>ej.kanban.Kanban({</br>allowKeyboard: true });</br>obj.appendTo("#kanban"); |`

| Settings | **Property:** `</br>keySettings</br></br>$("#Kanban").ejKanban({</br>keySettings: {</br>focus: "e",</br>insertCard: "45"</br>}});` | **Not Applicable** |

Toggle Columns

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** `</br>allowToggleColumn</br></br>$("#Kanban").ejKanban({</br>allowToggleColumn</br>: true});` | **Property:** `</br>allowToggle</br></br>var obj = new ej.kanban.Kanban({</br>columns:[{</br>allowToggle: true</br>}]);</br>obj.appendTo("#kanban");` |

| Toggle | **Method:** `toggleColumn</br>(headerText or $div)</br></br>var kanbanObj = $("#Kanban")</br>.data("ejKanban");</br>kanbanObj.toggleColumn</br>("Backlog");` | **Not Applicable** |

Dialog Editing

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Fields | **Property:** `editItems</br></br>$("#Kanban").ejKanban({</br>editSettings: {</br>editItems: [{}}});` | **Property:** `fields</br></br>var obj = new ej.kanban.Kanban({</br>dialogSettings: {</br>fields: [{}}});</br>obj.appendTo("#kanban");` |

| Dialog Model | **Not Available** | **Property:** `model</br></br>var obj = new </br>ej.kanban.Kanban({</br>dialogSettings: {</br>model: {</br>width: 250}}});</br>obj.appendTo("#kanban");` |

| Template | **Property:** `dialogTemplate</br></br>$("#Kanban").ejKanban({</br>editSettings: {</br>dialogTemplate: "#template" } });` | **Property:** `template</br></br>var obj = new ej.kanban.Kanban({</br>dialogSettings: {</br>template: </br>"#dialogTemplate"}});</br>obj.appendTo("#kanban");` |

| Enable Editing | **Property:** `allowEditing</br></br>$("#Kanban").ejKanban({</br>editSettings: {</br>allowEditing: true } });` | **In default allowed for editing.** |

| Enable Adding | **Property:** `allowAdding</br></br>$("#Kanban").ejKanban({</br>editSettings: {</br>allowAdding: true } });` | **Adding applicable using column</br> show add button or</br>public method.** |

| Edit Mode | **Property:** `editMode</br></br>$("#Kanban").ejKanban({</br>editSettings: {</br>editMode: ej.Kanban</br>.EditMode.Dialog } });` | **Not Applicable** |

| External Form template |

Property: `</br>externalFormTemplate</br></br>$("#Kanban").ejKanban({</br>editSettings: {</br>externalFormTemplate:</br>ej.Kanban.EditMode.Dialog</br>}});` | **Not Applicable** |

| External Form Position | **Property:**

`</br>externalFormPosition</br></br>$("#Kanban").ejKanban({</br>editSettings: {</br>externalFormPosition:</br>ej.Kanban.FormPosition.Bottom</br>}});` | **Not Applicable** |

| Add Card | **Method:** `</br>KanbanEdit.addCard</br>([primaryKey], [card])</br></br>var kanbanObj = $("#Kanban")</br>.data("ejKanban");</br>kanbanObj.KanbanEdit</br>.addCard("2",</br>{ Id: 2, Status: Open});` | **Method:** `</br></br>addCard(cardData)</br></br>var obj = new ej.kanban.Kanban({});</br>obj.appendTo("#kanban");</br>obj.addCard({ Id: 2,</br>Status: Open});` |

| Update Card | **Method:** *updateCard(key, data)*

```
var kanbanObj =
$("#Kanban").data("ejKanban");
kanbanObj.KanbanEdit.updateCard(2, { Id:
2, Status: Open});
```

| **Method:** *updateCard(cardData)*

```
var obj = new
ej.kanban.Kanban({});
obj.appendTo("#kanban");
obj.updateCard({ Id: 2, Status:
Open});
```

| Delete Card | **Method:** *KanbanEdit.deleteCard(key)*

```
var kanbanObj =
$("#Kanban").data("ejKanban");
kanbanObj.KanbanEdit.deleteCard(2);
```

| **Method:** *deleteCard()*

```
var obj = new
ej.kanban.Kanban({});
obj.appendTo("#kanban");
obj.deleteCard(2);
```

| Cancel Edit | **Method:** *KanbanEdit.cancelEdit()*

```
var kanbanObj =
$("#Kanban").data("ejKanban");
kanbanObj.KanbanEdit.cancelEdit();
```

| **Not Available** |

| End Edit | **Method:** *KanbanEdit.endEdit()*

```
var kanbanObj =
$("#Kanban").data("ejKanban");
kanbanObj.KanbanEdit.endEdit();
```

| **Not Available** |

| Start Edit | **Method:** *KanbanEdit.startEdit(\$div or key)*

```
var kanbanObj =
$("#Kanban").data("ejKanban");
kanbanObj.KanbanEdit.startEdit(2);
```

| **Method:** *openDialog(action, data)*

```
var obj = new
ej.kanban.Kanban({});
obj.appendTo("#kanban");
obj.openDialog("Add");
```

| Set Validation | **Method:** *KanbanEdit.setValidationToField(name, rules)*

```
var kanbanObj =
$("#Kanban").data("ejKanban");
kanbanObj.KanbanEdit.setValidationToField("Sum
mary", { required: true });
```

| **Not Available** |

| Close Dialog | **Not Applicable** | **Method:** *closeDialog()*

```
var obj = new
ej.kanban.Kanban({});
obj.appendTo("#kanban");
obj.closeDialog();
```

| Triggers before dialog Open | **Not Applicable** | **Event:** *dialogOpen*

```
var obj = new
ej.kanban.Kanban({
dialogOpen: function(args){}});
```

| Triggers when dialog close | **Not Applicable** | **Event:** *dialogClose*

```
var obj = new
ej.kanban.Kanban({
dialogClose: function(args){}});
```

| Triggers after the card is edited | **Event:** *endEdit*

```
$("#Kanban").ejKanban({
endEdit: function (args) {} });
```

| **Not Applicable** |

| Triggers after the card is deleted | **Event:** *endDelete*

```
$("#Kanban").ejKanban({
endDelete: function (args) {} });
```

| **Not Applicable** |

| Triggers before task is edited | **Event:** *beginEdit*

```
$("#Kanban").ejKanban({
beginEdit: function (args) {} });
```

| **Not Applicable** |

Dialog Editing Fields

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Fields | **Property:** *editItems*

```
$("#Kanban").ejKanban({
editSettings: {
editItems: [{]} });
```

| **Property:** *fields*

```
var obj = new ej.kanban.Kanban({
dialogSettings: {
fields: [{]} });
obj.appendTo("#kanban");
```

| Mapping key | **Property:** *field*
 {{{ "#Kanban").ejKanban({ editSettings: { editItems: { field: "Id" } } }); | **Property:** *key*
 {{{ "#Kanban").ejKanban({ dialogSettings: { fields: { key: "Id" } } }); obj.appendTo("#kanban"); |

| Label | **Not Applicable** | **Property:** *text*
 {{{ "#Kanban").ejKanban({ dialogSettings: { fields: { text: "ID", key: "Id" } } }); obj.appendTo("#kanban"); |

| Type | **Property:** *editType*
 {{{ "#Kanban").ejKanban({ editSettings: { editItems: { editType: ej.Kanban.EditingType.String } } }); | **Property:** *type*
 {{{ "#Kanban").ejKanban({ dialogSettings: { fields: { type: "TextBox", key: "Id" } } }); obj.appendTo("#kanban"); |

| Validation Rules | **Property:** *validationRules*
 {{{ "#Kanban").ejKanban({ editSettings: { editItems: { validationRules: { required: true } } } }); | **Property:** *validationRules*
 {{{ "#Kanban").ejKanban({ dialogSettings: { fields: { validationRules: { required: true } } } }); obj.appendTo("#kanban"); |

| Params | **Property:** *editParams*
 {{{ "#Kanban").ejKanban({ editSettings: { editItems: { editParams: { decimalPlaces: 2 } } } }); | **Not Applicable** |

| Default value | **Property:** *defaultValue*
 {{{ "#Kanban").ejKanban({ editSettings: { editItems: { defaultValue: "Open" } } }); | **Not Applicable** |

Tooltip

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** *tooltipSettings.enable*
 {{{ "#Kanban").ejKanban({ tooltipSettings: { enable: true } }); | **Property:** *enableTooltip*
 {{{ "#Kanban").ejKanban({ enableTooltip: true }); obj.appendTo("#kanban"); |

| Template | **Property:** *tooltipSettings.template*
 {{{ "#Kanban").ejKanban({ tooltipSettings: { template: "#tooltipTemplate" } }); | **Property:** *tooltipTemplate*
 {{{ "#Kanban").ejKanban({ tooltipTemplate: "#tooltipTemplate" }); obj.appendTo("#kanban"); |

Context Menu

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** *enable*
 {{{ "#Kanban").ejKanban({ contextMenuSettings: { enable: true } }); | **Not Applicable** |

| Menu Items | **Property:** *menuItems*
 {{{ "#Kanban").ejKanban({ contextMenuSettings: { enable: true, menuItems: ["Move Right"] } }); | **Not Applicable** |

| Disable default Items | **Property:** *disableDefaultItems*
 {{{ "#Kanban").ejKanban({ contextMenuSettings: { enable: true, disableDefaultItems: [ej.Kanban.MenuItem.MoveLeft] } }); | **Not Applicable** |

| Custom Menu Items | **Property:** *customMenuItems*
 {{{ "#Kanban").ejKanban({ contextMenuSettings: { enable: true, customMenuItems: { text: "Menu1" } } }); | **Target:** *target*
 {{{ "#Kanban").ejKanban({ contextMenuSettings: { enable: true, customMenuItems: { text: "Menu1" } } }); obj.appendTo("#kanban"); |

`{enable: true, customMenuItems: [{target: ej.Kanban.Target.Header }]};`
Template: `$("#Kanban").ejKanban({contextMenuSettings: {enable: true, customMenuItems: [{text: "Hide Column", template: "#template" }]};` | **Not Applicable** |

| Triggers when context menu item click | **Event:**
`contextClick` `$("#Kanban").ejKanban({contextClick: function (args) {};` | **Not Applicable** |

| Triggers when context menu open | **Event:**
`contextOpen` `$("#Kanban").ejKanban({contextOpen: function (args) {};` | **Not Applicable** |

WorkFlows

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** `workFlows` `$("#Kanban").ejKanban({workFlows: [{}];` | **Not Applicable** |

| Key | **Property:** `key` `$("#Kanban").ejKanban({workFlows: [{key: "Order"}];` | **Not Applicable** |

| Allowed Transition | **Property:** `allowedTransition` `$("#Kanban").ejKanban({workFlows: [{key: "Order", allowedTransitions: "Served"}];` | **Not Applicable** |

Filtering

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** `filterSettings` `$("#Kanban").ejKanban({filterSettings: [{}];` | **Not Applicable** |

| Enable | **Property:** `allowFiltering` `$("#Kanban").ejKanban({allowFiltering: true};` | **Not Applicable** |

| Text | **Property:** `text` `$("#Kanban").ejKanban({filterSettings: [{text: "Janet Issues"}];` | **Not Applicable** |

| Query | **Property:** `query` `$("#Kanban").ejKanban({filterSettings: [{query: new ej.Query().where("Assignee", "equal", "Janet")};` | **Not Applicable** |

| Description | **Property:** `description` `$("#Kanban").ejKanban({filterSettings: [{description: "Display Issues"}];` | **Not Applicable** |

| Filter Cards | **Method:** `filterCards(query)` `var kanbanObj = $("#Kanban").data("ejKanban"); kanbanObj.KanbanFilter.filterCards(new ej.Query().where("Assignee", "equal", "Janet"));` | **Not Applicable** |

| Clear | **Method:** `clearFilter()` `var kanbanObj = $("#Kanban").data("ejKanban"); kanbanObj.KanbanFilter.clearFilter();` | **Not Applicable** |

Searching

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** `</br>searchSettings</br></br>$("#Kanban").ejKanban({</br>searchSettings: []});` | **Not Applicable** |

| Enable | **Property:** `</br>allowSearching</br></br>$("#Kanban").ejKanban({</br>allowSearching: true</br>});` | **Not Applicable** |

| Fields | **Property:** `fields</br></br>$("#Kanban").ejKanban({</br>searchSettings: [{</br>fields: ["Summary"]}]);` | **Not Applicable** |

| Key | **Property:** `key</br></br>$("#Kanban").ejKanban({</br>searchSettings: [{</br>key: "Task 1"}]);` | **Not Applicable** |

| Operator | **Property:** `operator</br></br>$("#Kanban").ejKanban({</br>searchSettings: [{</br>operator: "contains"}]);` | **Not Applicable** |

| Ignore Case | **Property:** `ignoreCase</br></br>$("#Kanban").ejKanban({</br>searchSettings: [{</br>ignoreCase: true}]);` | **Not Applicable** |

| Search Cards | **Method:** `</br>searchCards(searchString)</br></br>var kanbanObj = $("#Kanban")</br>.data("ejKanban");</br>kanbanObj.KanbanFilter</br>.searchCards("Analyze");` | **Not Applicable** |

| Clear | **Method:** `clearSearch()</br></br>var kanbanObj = $("#Kanban")</br>.data("ejKanban");</br>kanbanObj.KanbanFilter</br>.clearSearch();` | **Not Applicable** |

External Drag And Drop

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | **Property:** `</br>allowExternalDragAndDrop</br></br>$("#Kanban").ejKanban({</br>allowExternalDragAndDrop</br>: true});` | **Not Applicable** |

| Target | **Property:** `</br>externalDropTarget</br></br>$("#Kanban").ejKanban({</br>cardSettings: {</br>externalDropTarget:</br>"#DroppedKanban" } });` | **Not Applicable** |

Scrolling

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | **Property:** `allowScrolling</br></br>$("#Kanban").ejKanban({</br>allowScrolling: true});` | **Not Applicable** |

| height | **Property:** `height</br></br>$("#Kanban").ejKanban({</br>allowScrolling: true,</br>scrollSettings: {</br>height: 400 } });` | **Property:** `height</br></br>var obj = new ej.kanban.Kanban({</br>height: 400});</br>obj.appendTo("#kanban");` |

| width | **Property:** `width</br></br>$("#Kanban").ejKanban({</br>allowScrolling: true,</br>scrollSettings: {</br>width: 400 } });` | **Property:** `width</br></br>var obj = new ej.kanban.Kanban({</br>width: 400});</br>obj.appendTo("#kanban");` |

| Freeze Swimlane | **Property:** `allowFreezeSwimlane</br></br>$("#Kanban").ejKanban({</br>allowScrolling: true,</br>scrollSettings: {</br>allowFreezeSwimlane: true</br>}});` | **Not Applicable** |

| Get Scroll Object | **Method:** `getScrollObject()`
`$("#Kanban").data("ejKanban");` | **Not Applicable** |

Card Selection and Hover

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Enable | **Property:** `allowSelection`
`$("#Kanban").ejKanban({allowSelection: true});` |
Property: `cardSettings.selectionType`
`var obj = new ej.kanban.Kanban({`
`cardSettings: {selectionType: "Single"};});` | **Not Applicable** |

| Type | **Property:** `selectionType`
`$("#Kanban").ejKanban({selectionType: "single"});` | It
 is covered under `selectionType` property. |

| Hover | **Property:** `allowHover`
`$("#Kanban").ejKanban({allowHover: true});` | **Not Applicable** |

| Clear | **Method:** `clear()`
`$("#Kanban").data("ejKanban").KanbanSelection.clear();` | **Not Applicable** |

| Triggers before card selected | **Event:**
`beforeCardSelect`
`$("#Kanban").ejKanban({beforeCardSelect: function (args)`
`{};});` | **Event:** `cardSelecting`
`$("#Kanban").ejKanban({cardSelecting: function`
`(args) {};});` | **Not Applicable** |

| Triggers after card selected | **Event:**
`cardSelect`
`$("#Kanban").ejKanban({cardSelect: function (args) {};});` | **Not Applicable** |

Toolbar

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Custom Toolbar | **Property:**
`customToolbarItems.template`
`$("#Kanban").ejKanban({customToolbarItems:`
`{template: "#Delete"};});` | **Not Applicable** |

| Triggers toolbar item click | **Event:**
`toolbarClick`
`$("#Kanban").ejKanban({toolbarClick: function (args) {};});` | **Not Applicable** |

Responsive

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** `isResponsive`
`$("#Kanban").ejKanban({isResponsive: true});` | **Not Applicable** |

| Minimum width | **Property:** `minWidth`
`$("#Kanban").ejKanban({isResponsive:`
`true,minWidth: 400});` | **Not Applicable** |

State Persistence

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Persistence | **Not Applicable** | **Property:** `enablePersistence`
var obj = new ej.kanban.Kanban({enablePersistence: true});obj.appendTo("#kanban"); |

Right to Left - RTL

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| default | **Property:** `enableRTL`
\$("#Kanban").ejKanban({enableRTL: true}) | **Property:** `enableRtl`
var obj = new ej.kanban.Kanban({enableRtl: true});obj.appendTo("#kanban"); |

Linear Gauge

Linear gauge dimensions in EJ2 JavaScript Linear gauge control

Size for Linear Gauge

The height and width of the Linear Gauge can be set using the [width](#) and [height](#) properties in Linear Gauge.

In Pixel

The size of the Linear Gauge can be set in pixel as demonstrated below.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  width: '650px',
  height: '350px'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```



```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In Percentage

By setting value in percentage, Linear Gauge receives its dimension matching to its parent. For example, when the height is set as **50%**, Linear Gauge renders to half of the parent height. The Linear Gauge will be responsive when the width is set as **100%**.

```

<div id='container'>
<div id='element' style="width:1000px; height:600px;"></div>
</div>

```

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  width: '100%',
  height: '50%'
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>

```



```
</body></html>
```

Note: When the component's size is not specified, the height will be **450px** and the width will be the same as the parent element's width.

Axis in EJ2 JavaScript Linear gauge control

Axis is used to indicate the numeric values in the linear scale. The Linear Gauge component can have any number of axes. The sub-elements of an axis are line, ticks, labels, ranges, and pointers.

Setting the start value and end value of the axis

The start value and end value for the Linear Gauge can be set using the [minimum](#) and [maximum](#) properties in the [axes](#) respectively. By default, the start value of the axis is **0** and the end value of the axis is **100**.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    minimum: 20,
    maximum: 140,
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Line Customization

The following properties in the [line](#) can be used to customize the axis line in the Linear Gauge.

- [height](#) - To set the length of the axis line.
- [width](#) - To set the thickness of the axis line.
- [color](#) - To set the color of the axis line.
- [offset](#) - To render the axis line with the specified distance from the Linear Gauge.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    line: {
      height: 150,
      width: 2,
      color: '#4286f4',
      offset: 2
    }
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Ticks Customization

Ticks are used to specify the interval in the axis. Ticks are of two types, major ticks and minor ticks. The following properties in the [majorTicks](#) and [minorTicks](#) can be used to customize the major ticks and minor ticks respectively

- [height](#) - To set the length of the major and minor ticks in pixel values.
- [color](#) - To set the color of the major and minor ticks of the Linear Gauge.
- [width](#) - To set the thickness of the major and minor ticks in pixel values.
- [interval](#) - To set the interval for the major ticks and minor ticks in the Linear Gauge.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    minimum: 20,
    maximum: 140,
    majorTicks: {
      interval: 20
    },
    minorTicks: {
      interval: 5,
      color: 'red'
    }
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Positioning the ticks

The minor and major ticks can be positioned by using [offset](#) and [position](#) properties. The [offset](#) is used to render the ticks with the specified distance from the axis. By default, the offset value is **0**. The possible values of the [position](#) property are **Inside**, **Outside**, **Cross**, and **Auto**. By default, the ticks will be placed inside the axis.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    minimum: 20,
    maximum: 140,
    majorTicks: {
      interval: 20,
      position: "Cross"
    },
    minorTicks: {
      interval: 5,
      color: 'red',
      position: "Outside",
      offset: 15
    }
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Labels Customization

The style of the labels can be customized using the following properties in the [font](#) property in [labelStyle](#).

- [color](#) - To set the color of the axis label.
- [fontFamily](#) - To set the font family of the axis label.
- [fontStyle](#) - To set the font style of the axis label.
- [fontWeight](#) - To set the font weight of the axis label.
- [opacity](#) - To set the opacity of the axis label.
- [size](#) - To set the size of the axis label.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    labelStyle: {
      font: {
        color: 'red'
      }
    }
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Positioning the axis label

Labels can be positioned by using [offset](#) and [position](#) properties in the [labelStyle](#). The [offset](#) defines the distance between the labels and ticks. By default, the offset value is 0. The possible values of the [position](#) property are **Inside**, **Outside**, **Cross**, and **Auto**. By default, labels will be placed inside the axis.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    labelStyle: {
      position: "Outside"
    }
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing the display of the last label

If the last label is not in the visible range, it will be hidden by default. The last label can be made visible by setting the [showLastLabel](#) property as **true** in the [axes](#).

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    showLastLabel: true,
    maximum: 115,
    line: {
      color: '#9E9E9E'
    },
    pointers: [{
      value: 20,
      height: 15,
      width: 15,
      color: '#757575',
      offset: 30
    }],
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Label Format

Axis labels in the Linear Gauge control can be formatted using the [format](#) property in the [labelStyle](#). It is used to render the axis labels in a certain format or to add a user-defined unit in the label. It works with the help of placeholder like {value}°C, where **value** represents the axis value. For example, 20°C.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes:[{
    labelStyle: {
      format: "{value}°C"
    }
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Displaying numeric format in labels

The numeric formats such as currency, percentage, and so on can be displayed in the labels of the Linear Gauge using the [format](#) property in the Linear Gauge component. The following table describes the result of applying some commonly used label formats on numeric values.

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The number is rounded to 1 decimal place.
1000	n2	1000.00	The number is rounded to 2 decimal place.
1000	n3	1000.000	The Number is rounded to 3 decimal place.

0.01	p1	1.0%	The number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The number is converted to percentage with 3 decimal place.
1000	c1	\$1,000.0	The currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	\$1,000.00	The currency symbol is appended to number and number is rounded to 2 decimal place.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    format: 'c',
    axes:[{
    }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Orientation

By default, the Linear Gauge is rendered vertically. To change its orientation, the [orientation](#) property must be set to **Horizontal**.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  orientation: "Horizontal",
  axes: [{
    minimum: 20,
    maximum: 140,
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Inverted Axis

The axis of the Linear Gauge component can be inverted by setting the [isInversed](#) property to **true** in the [axes](#).

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    isInversed: true
  }]
});
```

```
    }}
  }, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Opposed Axis

To place an axis opposite from its original position, [opposedPosition](#) property in the [axes](#) must be set as **true**.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    opposedPosition: true
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple Axes

Multiple axes can be added to the Linear Gauge by adding multiple [axis](#) object in the [axes](#) and customization can be done with the [axis](#). Each axis can be customized separately as shown in the following example.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    axes: [{
        labelStyle: {
            format: '{value}°C'
        }
    },
    {
        opposedPosition: true,
        labelStyle: {
            format: '{value}°F'
        }
    }
], '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Ranges in EJ2 JavaScript Linear gauge control

Range is the set of values in the axis. The range can be defined using the [start](#) and [end](#) properties in the [ranges](#). Any number of ranges can be added to the Linear Gauge using the [ranges](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    axes: [{
        ranges: [{
            start: 50,
            end: 80
        }]
    }]
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing the range

Ranges can be customized using the following properties in [ranges](#).

- [startWidth](#) - To set the thickness of the range at the start axis value.
- [endWidth](#) - To set the thickness of the range at the end axis value.
- [color](#) - Customize the range color.
- [position](#) - To place the range. By default, the range is placed outside of the axis. To change the position, this property can be set as "Inside", "Outside", "Cross", or "Auto".
- [Offset](#) - To place the range with specified distance from the axis.
- [border](#) - Customize color and width of range border.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    axes: [{
        ranges: [{
            start: 50,
            end: 80,
            startWidth: 10,
            endWidth: 20,
            color: 'red'
        }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="element"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Setting the range color of the labels

To set the color of the labels like the range color, set the [useRangeColor](#) property as **true** in the [labelStyle](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    ranges: [{
      start: 40,
      end: 80,
      startWidth: 10,
      endWidth: 10,
      color: 'red'
    }],
    labelStyle: {
      useRangeColor: true
    }
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple ranges

Multiple ranges can be added to the Linear Gauge by adding collections of [range](#) in the [ranges](#) and customization of ranges can be done with [range](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    axes: [{
        ranges: [
            {
                start: 0,
                end: 30,
                startWidth: 10,
                endWidth: 10,
                color: '#41f47f'
            },
            {
                start: 30,
                end: 50,
                startWidth: 10,
                endWidth: 10,
                color: '#f49441'
            },
            {
                start: 50,
                end: 100,
                startWidth: 10,
                endWidth: 10,
                color: '#cd41f4'
            }
        ]
    }
}],
    '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Gradient Color

Gradient support allows the addition of multiple colors in the range. The following gradient types are supported in the Linear Gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear-gradient, colors will be applied in a linear progression. The start value of the linear gradient can be set using the [startValue](#) property. The end value of the linear gradient will be set using the [endValue](#) property. The color stop values such as [color](#), [opacity](#), and [offset](#) to be defined in [colorStop](#).

INDEX.TS

```

import { LinearGauge, Gradient } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Gradient);
let gauge: LinearGauge = new LinearGauge({
    orientation: 'Horizontal',
    container: {
        width: 30, offset: 30
    },
    axes: [{
        line: { width: 0 },
        majorTicks: { height: 0, interval: 25 },
        minorTicks: { height: 0 },
        labelStyle: {
            font: { color: '#424242',
            }, offset: 55
        },
        pointers: [{
            value: 80, height: 25,
            width: 35, placement: 'Near',
            offset: -44, markerType: 'Triangle',
            color: '#f54ea2'
        }],
    }],

```

```

        ranges: [{
            start: 0, end: 80,
            startWidth: 30, endWidth: 30,
            offset: 30,
            linearGradient: {
                startValue: '0%',
                endValue: '100%',
                colorStop: [
                    { color: '#fef3f9', offset: '0%', opacity: 1 },
                    { color: '#f54ea2', offset: '100%', opacity: 1 }
                ]
            }
        }]
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [innerPosition](#) property. The outer circle position of the radial gradient can be set using the [outerPosition](#) property. The color stop values such as [color](#), [opacity](#), and [offset](#) to be defined in [colorStop](#).

INDEX.TS

```

import { LinearGauge, Gradient } from '@syncfusion/ej2-lineargauge';

```

```

LinearGauge.Inject(Gradient);
let gauge: LinearGauge = new LinearGauge({
  orientation: 'Horizontal',
  container: {
    width: 30, offset: 30
  },
  axes: [{
    line: { width: 0 },
    majorTicks: { height: 0, interval: 25 },
    minorTicks: { height: 0 },
    labelStyle: {
      font: { color: '#424242',
        }, offset: 55
    },
  },
  pointers: [{
    value: 80, height: 25,
    width: 35, placement: 'Near',
    offset: -44, markerType: 'Triangle',
    color: '#f54ea2'
  }],
  ranges: [{
    start: 0, end: 80,
    startWidth: 30, endWidth: 30,
    offset: 30,
    radialGradient: {
      radius: '65%',
      outerPosition: { x: '50%', y: '70%' },
      innerPosition: { x: '60%', y: '60%' },
      colorStop: [
        { color: '#fff5f5', offset: '5%', opacity: 0.9 },
        { color: '#f54ea2', offset: '100%', opacity: 0.9 }
      ]
    }
  }
]}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If we set both gradients for the range, only the linear gradient gets rendered. If we set the [startValue](#) and [endValue](#) of the [linearGradient](#) as empty strings, then the radial gradient gets rendered in the pointer of the Linear Gauge.

Pointers in EJ2 JavaScript Linear gauge control

Pointers are used to indicate values on an axis. The value of the pointer can be modified using the [value](#) property in [pointers](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    axes: [{
        pointers: [{
            value: 80
        }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Types of pointer

The Linear Gauge supports the following types of pointers:

- Bar
- Marker

The type of pointer can be modified by using the [type](#) property in [pointers](#).

Marker pointer

A marker pointer is a shape that can be used to mark the pointer value in the Linear Gauge.

Types of marker shapes

By default, the marker shape for the pointer is **InvertedTriangle**. To change the shape of the pointer, use the [markerType](#) property in [pointers](#). The following marker types are available in Linear Gauge.

- Circle
- Rectangle
- Triangle
- InvertedTriangle
- Diamond
- Image
- Text

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    axes: [{
        pointers: [{
            value: 80,
            type: 'Marker',
            markerType: 'Circle'
        }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Image can be rendered instead of rendering a shape as a pointer. It can be achieved by setting the [markerType](#) property to **Image** and setting the source URL of image to [imageUrl](#) property in [pointers](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge(
{
    orientation: 'Horizontal',
    axes: [
        {
            majorTicks: {
                interval: 20,
                position: 'Outside',
            },
            minorTicks: {
                position: 'Outside',
            },
            labelStyle: {
                position: 'Outside',
                font: { fontFamily: 'inherit' },
            },
            pointers: [
                {
                    value: 60,
                    markerType: 'Image',
                    imageUrl:
                        'https://ej2.syncfusion.com/demos/src/linear-
gauge/images/step-count.png',
                    offset: -27,
                    height: 40,
                    width: 40,
                },
            ],
        },
    ],
},
);

```

```
'#element'
);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Text can be added instead of rendering a shape as a pointer. It can be achieved by setting the [markerType](#) property to **Text**, and the text content can be set using the [text](#) property in [pointers](#).

The following properties in the [textStyle](#) property can be used to set the text style for the text pointer.

- [fontFamily](#) - It is used to set the font family for the text pointer.
- [fontStyle](#) - It is used to set the font style for the text pointer.
- [fontWeight](#) - It is used to set the font weight for the text pointer.
- [size](#) - It is used to set the font size for the text pointer.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge(
  {
    orientation: 'Horizontal',
    axes: [
      {
        line: {
          width: 5
        },
      },
    ],
  },
);
```

```
ranges: [
  {
    start: 0,
    end: 30,
    color: '#6FC78A',
    startWidth: 50,
    endWidth: 50,
    position: 'Inside'
  },
  {
    start: 30,
    end: 65,
    color: '#ECC85B',
    startWidth: 50,
    endWidth: 50,
    position: 'Inside'
  },
  {
    start: 65,
    end: 100,
    color: '#FB7D55',
    startWidth: 50,
    endWidth: 50,
    position: 'Inside'
  },
],
majorTicks: {
  interval: 20,
  height: 7,
  width: 1,
  position: 'Inside'
},
minorTicks: {
  interval: 10,
  height: 3,
  position: 'Inside'
},
labelStyle: {
  position: 'Inside',
  font: { fontFamily: 'inherit' }
},
pointers: [
  {
    value: 13,
    markerType: 'Text',
    text: 'Low',
    color: 'black',
    offset: '-50',
    textStyle: {
      size: '18px',
      fontWeight: 'bold'
    }
  },
  {
    value: 48,
    markerType: 'Text',
    text: 'Moderate',
  }
]
```



```

        color: 'black',
        offset: '-50',
        textStyle: {
            size: '18px',
            fontWeight: 'bold'
        }
    },
    {
        value: 83,
        markerType: 'Text',
        text: 'High',
        color: 'black',
        offset: '-50',
        textStyle: {
            size: '18px',
            fontWeight: 'bold'
        }
    },
    ],
    minimum: 0,
    maximum: 100,
    opposedPosition: true
}
]
},
'#element'
);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Marker Pointer Customization

The marker pointer can be customized using the following properties.

- [height](#) - To set the height of the pointer.
- [position](#) - The position of the pointer can be changed by setting the value as **Inside**, **Outside**, **Cross**, or **Auto**.
- [width](#) - To set the width of the pointer.
- [color](#) - To set the color of the pointer.
- [placement](#) - To place the pointer in the specified position. By default, the pointer is placed **Far** from the axis. To change the placement, set the [placement](#) property as **Near**, **Center**, or **None**.
- [offset](#) - To place the pointer with specified distance from the axis.
- [opacity](#) - To set the opacity of the pointer.
- [animationDuration](#) - To specify the duration of the animation in pointer.
- [border](#) - To set the color and width for the border of the pointer.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    pointers: [{
      value: 80,
      markerType: 'Circle',
      height: 15,
      width: 15,
      color: '#cd41f4'
    }]
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Bar Pointer

The bar pointer is used to track the axis value. The bar pointer starts from the beginning of the gauge and ends at the pointer value. To enable bar pointer set the [type](#) property property in [pointer](#) as Bar.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    axes: [{
        pointers: [{
            value: 60,
            type: 'Bar'
        }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Bar pointer customization

The bar pointer can be customized using following properties.

- [width](#) - To set the thickness of the bar pointer.
- [color](#) - To set the color of the bar pointer.
- [offset](#) - To place the bar pointer with the specified distance from it's default position.
- [opacity](#) - To set the opacity of the bar pointer.
- [roundedCornerRadius](#) - To set the corner radius for the bar pointer.
- [border](#) - To set the color and width for the border of the pointer.
- [animationDuration](#) - To set the duration of the animation in bar pointer.

Note: The placement property is not applicable for the bar pointer.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    pointers: [{
      value: 60,
      type: 'Bar',
      width: 20,
      color: '#f44141'
    }]
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple pointers

Multiple pointers can be added to the Linear Gauge by adding multiple [pointer](#) objects in the [pointers](#) and customization for the pointers can be done with the [pointer](#) object.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    pointers: [
      {
        value: 80
      },
      {
        value: 60,
        markerType: 'Circle'
      },
      {
        value: 30,
        markerType: 'Diamond'
      }
    ]
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Pointer animation

Pointer is animated on loading the gauge. This can be handled using the [animationDuration](#) property. The duration of the animation can be specified in milliseconds.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    pointers: [{
      value: 60,
      animationDuration: 1000
    }]
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Gradient Color

Gradient support allows the addition of multiple colors in the pointers of the Linear Gauge. The following gradient types are supported in the Linear Gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear gradient, colors will be applied in a linear progression. The start value of the linear gradient can be set using the [startValue](#) property. The end value of the linear gradient will be set using the [endValue](#) property. The color stop values such as [color](#), [opacity](#), and [offset](#) are set using [colorStop](#) property. The linear gradient can be rendered for the pointer in the Linear Gauge by using the below example.

INDEX.TS

```
import { LinearGauge, Gradient } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Gradient);
let gauge: LinearGauge = new LinearGauge({
  orientation: 'Horizontal',
  container: {
    width: 30, offset: 30
  },
  axes: [{
    line: { width: 0 },
    majorTicks: { height: 0, interval: 25 },
    minorTicks: { height: 0 },
    labelStyle: {
      font: { color: '#424242',
        }, offset: 55
    },
  },
  pointers: [{
    value: 80, height: 25,
    width: 35, placement: 'Near',
    offset: -44, markerType: 'Triangle',
    color : '#f54ea2',
    linearGradient: {
      startValue: '0%',
      endValue: '100%',
      colorStop: [
        { color: '#fef3f9', offset: '0%', opacity: 1 },
        { color: '#f54ea2', offset: '100%', opacity: 1 }]
    }
  }],
  ranges: [{
    start: 0, end: 80,
    startWidth: 30, endWidth: 30,
    color: '#f54ea2', offset: 30,
  }]
}],
'#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [innerPosition](#) property. The outer circle position of the radial gradient can be set using the [outerPosition](#) property. The color stop values such as [color](#), [opacity](#), and [offset](#) are set using [colorStop](#) property. The radial gradient can be rendered for the pointer in the Linear Gauge by using the below example.

INDEX.TS

```

import { LinearGauge, Gradient } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Gradient);
let gauge: LinearGauge = new LinearGauge({
    orientation: 'Horizontal',
    container: {
        width: 30, offset: 30
    },
    axes: [{
        line: { width: 0 },
        majorTicks: { height: 0, interval: 25 },
        minorTicks: { height: 0 },
        labelStyle: {
            font: { color: '#424242',
            }, offset: 55
        },
    },
    pointers: [{
        value: 80, height: 25,
        width: 35, placement: 'Near',
        offset: -44, markerType: 'Triangle',
        color : '#f54ea2',
        radialGradient: {
            radius: '60%',
            outerPosition: { x: '50%', y: '50%' },

```



```

        innerPosition: { x: '50%', y: '50%' },
        colorStop: [
            { color: '#fff5f5', offset: '0%', opacity: 0.9 },
            { color: '#f54ea2', offset: '100%', opacity: 0.8 }
        ]
    }
},
ranges: [{
    start: 0, end: 80,
    startWidth: 30, endWidth: 30,
    color: '#f54ea2', offset: 30,
}]
}]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: If we set both gradients, only the linear gradient gets rendered. If we set the [startValue](#) and [endValue](#) of the [linearGradient](#) as empty strings, then the radial gradient gets rendered in the pointer of the Linear Gauge.

Annotations in EJ2 JavaScript Linear gauge control

Annotations are used to mark the specific area of interest in the Linear Gauge with text, HTML elements, or images. Any number of annotations can be added to the Linear Gauge component.

Adding annotation

To render the custom HTML elements in the Linear Gauge component, use the [content](#) property in the [annotations](#). The annotation can be rendered either by specifying the id of the element or specifying the code to create a new element that needs to be displayed in the gauge area.

```
,
<script id='fruits' type='text/x-template'>
<div id='apple'>
<img src='src/lineargauge/images/apple.png'>
</div>
</script>
,
`ts
import { LinearGauge, Annotations } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Annotations);
let gauge: LinearGauge = new LinearGauge({
  annotations: [{
    content: '#fruits',
    x: 100,
    y: 100,
    zIndex: "1"
  }]
}, '#element');
,
```

Customization

The following properties are used to customize the annotation.

- [zIndex](#) - Bring the annotation to the front or back, when annotation overlaps with another element.
- [axisValue](#) - To place the annotation in the specified axis value with respect to the provided axis index.
- [axisIndex](#) - To place the annotation in the specified axis with respect to the provided axis value.
- [horizontalAlignment](#) - To place the annotation horizontally.
- [verticalAlignment](#) - To place the annotation vertically.
- [x](#), [y](#) - To place the annotation in the specified location.

Changing the z-index

To change the stack order of an annotation element, the [zIndex](#) property of the [annotations](#) can be used.

INDEX.TS

```
import { LinearGauge, Annotations } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Annotations);
let gauge: LinearGauge = new LinearGauge({
  annotations: [{
    content: '<div id="first"><h1>Gauge</h1></div>',
    verticalAlignment: 'Center',
    horizontalAlignment: 'Center',
    zIndex: '1'
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Positioning an annotation

The annotation can be placed anywhere in the Linear Gauge by setting the pixel value to the [x](#) and [y](#) properties in the [annotations](#).

INDEX.TS

```
import { LinearGauge, Annotations } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Annotations);
let gauge: LinearGauge = new LinearGauge({
  annotations: [{
    content: '<div id="first"><h1>Gauge</h1></div>',
    x: 100,
    y: 100,
```

```

        zIndex: '1'
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Alignment of annotation

The annotation can be aligned horizontally and vertically by using [horizontalAlignment](#) and [verticalAlignment](#) properties respectively. The possible values can be **Center**, **Far**, **Near**, and **None**. The [horizontalAlignment](#) and [verticalAlignment](#) properties are not applicable when the [x](#) and [y](#) properties are set in the [annotations](#).

INDEX.TS

```

import { LinearGauge, Annotations } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Annotations);
let gauge: LinearGauge = new LinearGauge({
  annotations: [{
    content: '<div id="first"><h1>Gauge</h1></div>',
    verticalAlignment: 'Center',
    horizontalAlignment: 'Center',
    zIndex: '1'
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple annotations

Multiple annotations can be added to the Linear Gauge component by adding the multiple [annotation](#) in the [annotations](#) and customization for the annotation can be done with the [annotation](#).

INDEX.TS

```

import { LinearGauge, Annotations } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Annotations);
let gauge: LinearGauge = new LinearGauge({
  annotations: [
    {
      content: '<div><h1 style="color:red;">Speed</h1></div>',
      verticalAlignment: 'Near',
      horizontalAlignment: 'Center',
      x: 100,
      y: 150,
      zIndex: '1'
    },
    {
      content: '<div><h1 style="color:blue;">Meter</h1></div>',
      verticalAlignment: 'Center',
      horizontalAlignment: 'Center',
      x: -100,
      y: -100,
      zIndex: '1'
    }
  ]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Animation in EJ2 JavaScript Linear Gauge control

All of the elements in the Linear Gauge, such as the axis lines, ticks, labels, ranges, pointers, and annotations, can be animated sequentially by using the [animationDuration](#) property. The animation for the Linear Gauge is enabled when the [animationDuration](#) property is set to an appropriate value in milliseconds, providing a smooth rendering effect for the control. If the [animationDuration](#) property is set to **0**, which is the default value, the animation effect is disabled. If the animation is enabled, the control will behave in the following order.

1. The axis line, ticks, labels, and ranges will all be animated at the same time.
2. If available, pointers will be animated in the same way as [pointer animation](#).
3. If available, annotations will be animated.

The animation of the Linear Gauge is demonstrated in the following example.

INDEX.TS

```

import { LinearGauge, Annotations } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Annotations);
let gauge: LinearGauge = new LinearGauge({
  animationDuration: 3000,
  orientation: 'Horizontal',
  axes: [
    {
      pointers: [

```

```

        {
            value: 10,
            height: 15,
            width: 15,
            placement: 'Near',
            offset: -38,
            markerType: 'Triangle',
        },
    ],
    ranges: [
        {
            start: 0,
            end: 50,
            startWidth: 10,
            endWidth: 10,
            color: '#F45656',
            offset: 35,
        },
    ],
    majorTicks: {
        interval: 10,
        height: 20,
        color: '#9E9E9E',
    },
    minorTicks: {
        interval: 2,
        height: 10,
        color: '#9E9E9E',
    },
    labelStyle: {
        offset: 48,
        font: {
            fontFamily: 'inherit',
        },
    },
    ],
    annotations: [
        {
            content:
                '<div id="pointer" style="width:70px;margin-left:-3%;margin-top: 21%;font-size:16px;">10 MPH</div>',
            axisIndex: 0,
            axisValue: 10,
            x: 10,
            y: -70,
            zIndex: '1',
        },
    ],
    });
gauge.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

User interaction in EJ2 JavaScript Linear gauge control

Tooltip

Linear gauge displays the details about a pointer value through [tooltip](#), when the mouse hovers over the pointer. To enable the tooltip, set the [enable](#) property to **true** and inject the **GaugeTooltip** module in Linear Gauge.

INDEX.TS

```

import { LinearGauge, GaugeTooltip } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(GaugeTooltip);
let gauge: LinearGauge = new LinearGauge({
    tooltip: {
        enable: true
    },
    axes: [{
        pointers: [{
            value: 80
        }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">

```



```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lineargauge/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip Format

Tooltip in the Linear Gauge control can be formatted using the [format](#) property in [tooltip](#) object. It is used to render the tooltip in certain format or to add a user-defined unit in the tooltip. By default, the tooltip shows the pointer value only. In addition to that, more information can be added in the tooltip. For example, the format **{value}km** shows pointer value with kilometer unit in the tooltip.

INDEX.TS

```

import { LinearGauge, GaugeTooltip } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(GaugeTooltip);
let gauge: LinearGauge = new LinearGauge({
    tooltip: {
        enable: true,
        format: '{value}km'
    },
    axes: [{
        pointers: [{
            value: 80
        }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lineargauge/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip Template

The HTML element can be rendered in the tooltip of the Linear Gauge using the [template](#) property in the [tooltip](#). The `${value}` can be used as placeholders in the HTML element to display the pointer values of the corresponding axis.

INDEX.TS

```

import { LinearGauge, GaugeTooltip } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(GaugeTooltip);
let gauge: LinearGauge = new LinearGauge({
    tooltip: {
        enable: true,
        //tooltip template for Linear gauge
        template: '<div>Pointer: 80 </div>'
    },
    axes: [{
        pointers: [{
            value: 80
        }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lineargauge/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the appearance of the tooltip

The tooltip can be customized using the following properties in [tooltip](#).

- [fill](#) - To fill the color for tooltip.
- [enableAnimation](#) - To enable or disable the tooltip animation.
- [border](#) - To set the border color and width of the tooltip.
- [textStyle](#) - To customize the style of the text in tooltip.
- [showAtMousePosition](#) - To show the tooltip at the mouse position.

INDEX.TS

```

import { LinearGauge, GaugeTooltip } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(GaugeTooltip);
let gauge: LinearGauge = new LinearGauge({
    tooltip: {
        enable: true,
        fill: '#e5bcbc',
        border: {
            color: '#d80000',
            width: 2
        }
    },
    axes: [{
        pointers: [{
            value: 80
        }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lineargauge/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Positioning the tooltip

The tooltip is positioned at the **End** of the pointer. To change the position of the tooltip at the start, or center of the pointer, set the [position](#) property to **Start** or **Center**.

INDEX.TS

```

import { LinearGauge, GaugeTooltip } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(GaugeTooltip);
let gauge: LinearGauge = new LinearGauge({
  tooltip: {
    enable: true,
    position: "Center"
  },
  axes: [{
    pointers: [{
      type: 'Bar',
      value: 50,
      offset: -50,
      color: "red"
    }],
    majorTicks: {
      interval: 10
    },
    minorTicks: {
      interval: 2
    },
  }],
});

```

```

        labelStyle: {
            offset: 48
        },
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lineargauge/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Pointer Drag

To drag either the marker or bar pointer to the desired axis value, set the [enableDrag](#) property as **true** in the [pointer](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    pointers: [{
      value: 80,
      enableDrag: true
    }]
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lineargauge/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Linear gauge print and export in EJ2 JavaScript Linear gauge control

Print

The rendered linear gauge can be printed directly from the browser by calling the [print](#) method. To use the print functionality, set the [allowPrint](#) property as **true**.

INDEX.JS

```

var gauge = new ej.lineargauge.LinearGauge({
  allowPrint: true
}, '#element');
document.getElementById('print').onclick = () => {
  gauge.print();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 20%;">
        <div id="element" style="margin-top: 20%;"></div>
        <button id="print" type="button" width="15%" style="float:
right">Print</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export

Image Export

To use the image export functionality, set the [allowImageExport](#) property as **true**. The rendered linear gauge can be exported as an image using the [export](#) method. This method requires two parameters: export type and file name. The Linear Gauge can be exported as an image with the following formats.

- JPEG
- PNG
- SVG

INDEX.JS

```

var gauge = new ej.linearGauge.LinearGauge({
    allowImageExport: true
}, '#element');
document.getElementById('export').onclick = () => {
    gauge.export("PNG", "Gauge");
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 20%;">
        <div id="element" style="margin-top: 20%;"></div>
        <button id="export" type="button" width="15%" style="float:
right">Export</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

PDF Export

To use the PDF export functionality, set the [allowPdfExport](#) property as **true**. The rendered Linear Gauge can be exported as PDF using the [export](#) method. The [export](#) method requires three parameters: file type, file name, and orientation of the PDF document. The orientation of the PDF document can be set as **Portrait** or **Landscape**.

INDEX.JS

```

var gauge = new ej.linearGauge.LinearGauge({
    allowPdfExport: true
}, '#element');
document.getElementById('export').onclick = () => {
    gauge.export("PDF", "Gauge", 0);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 20%;">

```



```

        <div id="element" style="margin-top: 20%;"></div>
        <button id="export" type="button" width="15%" style="float:
right">Export</button>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting Linear Gauge as base64 string of the file

The Linear Gauge can be exported as base64 string for the JPEG, PNG and PDF formats. The rendered Linear Gauge can be exported as base64 string of the exported image or PDF document used in the [export](#) method. The arguments that are required for this method is export type, file name, orientation of the exported PDF document and **allowDownload** boolean value that is set as **false** to return base64 string. The value for the orientation of the exported PDF document is set as **null** for image export and **Portrait** or **Landscape** for the PDF document.

INDEX.JS

```

var gauge = new ej.lineargauge.LinearGauge({
    allowImageExport: true
}, '#element');
document.getElementById('export').onclick = () => {
    gauge.export('JPEG', 'Gauge', null, false).then((data) => {
        var base64 = data;
        document.writeln(base64);
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 20%;">
        <div id="element" style="margin-top: 20%;"></div>
        <button id="export" type="button" width="15%" style="float:
right">Export</button>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The exporting of the Linear Gauge as base64 string is not applicable for the SVG format.

Linear gauge appearance in EJ2 JavaScript Linear gauge control

Customizing the Linear Gauge area

The following properties are available in the [LinearGauge](#) to customize the Linear Gauge area.

- [background](#) - Applies the background color for the Linear Gauge.
- [border](#) - To customize the color and width of the border in Linear Gauge.
- [margin](#) - To customize the margins of the Linear Gauge.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    background: 'skyblue',
    border: {
        color: "#FF0000",
        width: 2
    },
    margin: {
        left: 40,
        right: 40,
        top: 40,
        bottom: 40
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

```

```
<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Setting up the Linear Gauge title

The title for the Linear Gauge can be set using [title](#) property in Linear Gauge. Its appearance can be customized using the [titleStyle](#) with the below properties.

- [color](#) - Specifies the text color of the title.
- [fontFamily](#) - Specifies the font family of the title.
- [fontStyle](#) - Specifies the font style of the title.
- [fontWeight](#) - Specifies the font weight of the title.
- [opacity](#) - Specifies the opacity of the title.
- [size](#) - Specifies the font size of the title.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  // Title for linear gauge.
  title: 'linear gauge',
  titleStyle: {
    fontFamily: 'Arial',
    fontStyle: 'italic',
    fontWeight: 'regular',
    color: '#E27F2D',
    size: '23px'
  }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing the Linear Gauge container

The area used to render the ranges and pointers at the center position of the gauge is called container. The following types of container to be applicable for Linear Gauge.

- Normal
- Rounded Rectangle
- Thermometer

The type of the container can be modified by using the [type](#) property in [container](#). The container can be customized by using the following properties in [container](#).

- [offset](#) - To place the container with the specified distance from the axis of the Linear Gauge.
- [width](#) - To set the thickness of the container.
- [height](#) - To set the length of the container.
- [backgroundColor](#) - To set the background color of the container.
- [border](#) - To set the color and width for the border of the container.

Normal

The **Normal** type will render the container as a rectangle and this is the default container type.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    container: {
        width: 30
    },
    axes: [{
        pointers: [{
            value: 50,
            width: 15,
            type: 'Bar'
        }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Rounded Rectangle

The **RoundedRectangle** type will render the container as a rectangle with rounded corner radius. The rounded corner radius of the container can be customized using the [roundedCornerRadius](#) property in [container](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  container: {
    width: 30,
    type: 'RoundedRectangle'
  },
  axes: [{
    pointers: [{
      value: 50,
      width: 15,
      type: 'Bar'
    }]
  }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Thermometer

The **Thermometer** type will render the container similar to the appearance of thermometer.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    container: {
        width: 30,
        type: 'Thermometer'
    },
    axes: [{
        pointers: [{
            value: 50,
            width: 15,
            type: 'Bar'
        }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Fitting the Linear Gauge to the control

The Linear Gauge component is rendered with margin by default. To remove the margin around the Linear Gauge, the [allowMargin](#) property in [LinearGauge](#) is set as **false**.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    allowMargin: false,
    background: "skyblue",
    border: {
        width: 3,
        color: "red"
    },
    margin: {
        left: 0,
        right: 0,
        top: 0,
        bottom: 0
    },
    axes: [{
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To use this feature, set the [allowMargin](#) property to **false**, the [width](#) property to **100%** and the properties of [margin](#) to **0**.

Accessibility in EJ2 JavaScript Linear Gauge control

Linear Gauge has built-in accessibility features like screen reading and WAI-ARIA attributes.

WAI-ARIA attributes

The Linear Gauge control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Linear Gauge control:

Attributes	Purpose
role=region	It is specified in the pointer where the interactive drag and drop function is supported to update the pointer value.
aria-label	Provides an accessible name for the axis labels, text pointer and annotation.

Screen reading in Linear Gauge

Accessibility in the Linear Gauge control ensures that all users, regardless of ability or disability, can use screen reading. The following Linear Gauge elements will be read aloud using screen reading software, such as Narrator for Windows.

Elements	Description
Axis labels	Reads the axis labels of the Linear Gauge.
Text pointer	Reads the text content shown as a pointer in Linear Gauge.
Annotation	Reads the content specified in the annotation.

Ensuring accessibility

The Linear Gauge control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Linear Gauge control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Linear Gauge control with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript controls](#)

Internationalization in EJ2 JavaScript Linear gauge control

Globalization is the process of designing and developing a component that works in different cultures. Internationalization is used to globalize the number content in Linear Gauge component using [format](#) property in Linear Gauge component. It has static text on some features such as

- Axis label
- Tooltip

The static text on above features can be changed to any culture such as Arabic, Deutsch and French. To know more about the globalization in JavaScript components, refer [here](#)

Numeric Format

The text in axis labels and tooltip can be displayed in the numeric format such as currency, percentage and so on. To know more about the numeric formats in axis labels, refer [here](#). In the below example, the axis label is displayed in the currency format.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
import { loadCldr, setCulture, Ajax, setCurrencyCode } from
 '@syncfusion/ej2-base';
setCulture("de");
setCurrencyCode('EUR');
let gauge: LinearGauge = new LinearGauge({
  locale: "de",
  format: 'c',
  axes: [{
    labelStyle: {
      font: {
        color: 'red'
      }
    }
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Events in EJ2 JavaScript Linear gauge control

This section describes the Linear Gauge component's event that gets triggered when corresponding operations are performed.

animationComplete

When the pointer animation is completed, the [animationComplete](#) event will be triggered. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    animationComplete: function() {
    },
    axes:[{
        pointers:[{
            value: 10
        }]
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 20%;">
        <div id="element" style="margin-top: 20%;"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

annotationRender

Before the annotation is rendered in the Linear Gauge, the [annotationRender](#) event will be triggered. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```

import { LinearGauge, Annotations } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Annotations);
let gauge: LinearGauge = new LinearGauge({
    annotationRender: function() {
    },
    annotations: [{
        content: '<div id="first"><h1>Gauge</h1></div>',
        verticalAlignment: 'Center',
        horizontalAlignment: 'Center',
        zIndex: '1'
    }]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 20%;">

```

```

        <div id="element" style="margin-top: 20%;"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

axisLabelRender

Before each axis label is rendered in the Linear Gauge, the [axisLabelRender](#) event is fired. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    axisLabelRender: function() {
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 20%;">
        <div id="element" style="margin-top: 20%;"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

beforePrint

The [beforePrint](#) event is fired before the print begins. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```
import { LinearGauge, Print } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Print);
let gauge: LinearGauge = new LinearGauge({
  allowPrint: true,
  beforePrint: function() {
  }
}, '#element');
document.getElementById('print').onclick = () => {
  gauge.print();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
    <button id="print" type="button" width="15%" style="float:
right">Print</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

dragEnd

The [dragEnd](#) event will be fired before the pointer drag is completed. To know more about the argument of this event, refer [here](#).

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  dragEnd: function() {
  },
  axes : [{
    pointers: [{
      enableDrag: true
    }]
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

dragMove

The [dragMove](#) event will be fired when the pointer is dragged. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  dragMove: function() {
  },
  axes : [{
    pointers: [{
      enableDrag: true
    }]
  }]
});
```

```
    }]
  }, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

dragStart

When the pointer drag begins, the [dragStart](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  dragStart: function() {
  },
  axes : [{
    pointers: [{
      enableDrag: true
    }]
  }]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 20%;">
        <div id="element" style="margin-top: 20%;"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

gaugeMouseDown

When mouse is pressed down on the gauge, the [gaugeMouseDown](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    gaugeMouseDown: function() {
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

    <div id="container" style="margin-top: 20%;">
      <div id="element" style="margin-top: 20%;"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

gaugeMouseLeave

When mouse pointer moves over the gauge, the [gaugemouseleave](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  gaugeMouseLeave: function() {
  }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

gaugeMouseMove

When mouse pointer leaves the gauge, the [gaugeMouseMove](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  gaugeMouseMove: function() {
  }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

gaugeMouseUp

When the mouse pointer is released over the Linear Gauge, the [gaugeMouseUp](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  gaugeMouseUp: function() {
  }
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

load

Before the Linear Gauge is loaded, the [load](#) event is fired. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  load: function() {
  }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container" style="margin-top: 20%;">
        <div id="element" style="margin-top: 20%;"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

loaded

After the Linear Gauge has been loaded, the [loaded](#) event will be triggered. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
    loaded: function() {
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 20%;">
        <div id="element" style="margin-top: 20%;"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

resized

After the window resizing, the [resized](#) event is triggered. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  resized: function() {
  }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

tooltipRender

The [tooltipRender](#) event is fired before the tooltip is rendered. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```
import { LinearGauge, GaugeTooltip } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(GaugeTooltip);
let gauge: LinearGauge = new LinearGauge({
  tooltipRender: function() {
  },
```

```

tooltip: {
  enable: true
},
axes: [{
  pointers: [{
    value: 40
  }]
}]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

valueChange

The [valueChange](#) event is triggered when the pointer is dragged from one value to another. To know more about the arguments of this event, refer [here](#).

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  valueChange: function() {
  },
  axes: [{
    pointers: [{
      enableDrag: true,
      value: 40
    }]
  }]
});

```

```

    }]
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Methods in EJ2 JavaScript Linear gauge control

The following methods are available in the Linear Gauge component.

setPointerValue

To change the pointer value dynamically, use the [setPointerValue](#) method in the Linear Gauge component. The following are the arguments for this method.

Argument name	Description
axisIndex	Specifies the index of the axis in which the pointer value is to be updated.
pointerIndex	Specifies the index of the pointer to be updated.
pointerValue	Specifies the value of the pointer to be updated.

INDEX.TS

```

import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes: [{
    pointers: [{
      value: 80
    }]
  }]
});

```

```

    }}
  }, '#element');
document.getElementById('btn').onclick = () => {
  gauge.setPointerValue(0, 0, 30);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
    <button id="btn" type="button" width="15%" style="float:
right">Click me</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

setAnnotationValue

To change the annotation content dynamically, use the [setAnnotationValue](#) method in the Linear Gauge component. The following are the arguments for this method.

Argument name	Description
-----	-----
annotationIndex	Specifies the index number of the annotation to be updated.
content	Specifies the text for the annotation to be updated.
axisValue	Specifies the value of the axis where the annotation is to be placed.

INDEX.TS

```

import { LinearGauge, Annotations } from '@syncfusion/ej2-lineargauge';
LinearGauge.Inject(Annotations);

```



```
let gauge: LinearGauge = new LinearGauge({
  annotations: [{
    content: '10',
    zIndex: '1',
    axisValue: 0
  }]
}, '#element');
document.getElementById('btn').onclick = () => {
  gauge.setAnnotationValue(0, '50', 50);
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
    <button id="btn" type="button" width="15%" style="float:
right">Click me</button>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

refresh

The [refresh](#) method can be used to change the state of the component and render it again.

INDEX.TS

```
import { LinearGauge } from '@syncfusion/ej2-lineargauge';
let gauge: LinearGauge = new LinearGauge({
  axes:[{
    pointers:[{
      value: 10
    }]
  }]
}, '#element');
```

```
document.getElementById('btn').onclick = () => {
    gauge.refresh();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 20%;">
    <div id="element" style="margin-top: 20%;"></div>
    <button id="btn" type="button" width="15%" style="float:
right">Click me</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Ej1 api migration in EJ2 JavaScript Linear gauge control

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

Linear gauge dimensions

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Height | **Property:** *height*

 \$("#container").ejLinearGauge({ height: 150 }); | **Property:** *height*

 let gauge: LinearGauge = new LinearGauge({
 height : '150px'
 });
 gauge.appendTo('#container'); |

| Width | **Property:** *width*

 \$("#container").ejLinearGauge({ width: 200 }); | **Property:** *width*

 let gauge: LinearGauge = new LinearGauge({
 width : '200px'
 });
 gauge.appendTo('#container'); |

| Height(In Percentage) | Not Applicable | **Property:** *height*

 let gauge: LinearGauge = new LinearGauge({
 height : '70%'
 });
 gauge.appendTo('#container'); |

|Width(In Percentage)| Not Applicable | **Property:** *width*
 let gauge: LinearGauge = new LinearGauge({ width : '100%' }); gauge.appendTo('#container');

Line customizaton

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Height| **Property:** *scales.length*
 \$("#container").ejLinearGauge({ scales:[{ length: 300 }]});
Property: *axes.line.height*
 let gauge: LinearGauge = new LinearGauge({ axes: [{ line: { height : 150 } }]}); gauge.appendTo('#container');

|Width| **Property:** *scales.width*
 \$("#container").ejLinearGauge({ scales:[{ width: 300 }]});
Property: *axes.line.width*
 let gauge: LinearGauge = new LinearGauge({ axes: [{ line: { width : 2 } }]}); gauge.appendTo('#container');

|Color| **Property:** *scales.backgroundColor*
 \$("#container").ejLinearGauge({ scales:[{ backgroundColor: "blue" }]});
Property: *axes.line.color*
 let gauge: LinearGauge = new LinearGauge({ axes: [{ line: { color : '#4286f4' } }]}); gauge.appendTo('#container');

|Offset| Not Applicable | **Property:** *axes.line.offset*
 let gauge: LinearGauge = new LinearGauge({ axes: [{ line: { offset : 2 } }]}); gauge.appendTo('#container');

|Opacity| **Property:** *scales.opacity*
 \$("#container").ejLinearGauge({ scales:[{ opacity: 0.2 }]}); | Not Applicable |

|DashArray| Not Applicable | **Property:** *axes.line.dashArray*
 let gauge: LinearGauge = new LinearGauge({ axes: [{ line: { dashArray : '1' } }]}); gauge.appendTo('#container');

Ticks

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Type of Ticks| **Property:** *scales.ticks.type*
 \$("#container").ejLinearGauge({ scales:[{ ticks: [{ type: "majorinterval" }] }]});
Property: *axes.majorTicks.height*
 let gauge: LinearGauge = new LinearGauge({ axes: [{ majorTicks: { } }]}); gauge.appendTo('#container');

|Height of Major Ticks| **Property:** *scales.ticks.height*
 \$("#container").ejLinearGauge({ scales: [{ ticks: [{ type: "majorinterval", height: 8 }] }]});
Property: *axes.majorTicks.height*
 let gauge: LinearGauge = new LinearGauge({ axes: [{ majorTicks: { height : 10 } }]}); gauge.appendTo('#container');

|Width of Major Ticks| **Property:** *scales.ticks.width*
 \$("#container").ejLinearGauge({ scales:[{ ticks: [{ type: "majorinterval", width: 5 }] }]});
Property: *axes.majorTicks.width*
 let gauge: LinearGauge = new LinearGauge({ axes: [{ majorTicks: { width : 2 } }]}); gauge.appendTo('#container');

| Color of Major Ticks | **Property:** *scales.ticks.color*
 \$("#container").ejLinearGauge({
 scales:[
 ticks: [{ type: "majorinterval", color: "Blue" }]
 }]);
Property: *axes.majorTicks.color*
 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 majorTicks: { color: "Blue: }
 }];
 gauge.appendTo('#container');

| Offset for Major Ticks | **Property:** *scales.ticks.distanceFromScale*
 \$("#container").ejLinearGauge({
 scales:[
 ticks: [{ distanceFromScale:
 { x: 5, y: 5 }, type: "majorinterval" }]
 }]);
Property: *axes.majorTicks.offset*
 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 majorTicks: { offset
 : 1 }
 }];
 gauge.appendTo('#container');

| Interval of Major Ticks | **Property:** *scales.majorIntervalValue*
 \$("#container").ejLinearGauge({
 scales: [{ majorIntervalValue: 15 }]
 }]);
Property: *axes.majorTicks.interval*
 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 majorTicks: { interval : 20 }
 }];
 gauge.appendTo('#container');

| Angle of Major Ticks | **Property:** *scales.ticks.angle*
 \$("#container").ejLinearGauge({
 scales:[
 ticks: [{ type: "majorinterval", angle: 30 }]
 }]);
 Not Applicable

| Opcity of Major Ticks | **Property:** *scales.ticks.opacity*
 \$("#container").ejLinearGauge({
 scales:[
 ticks: [{ type: "majorinterval", opacity: 0.5 }]
 }]);
 Not Applicable

| Height of Minor Ticks | **Property:** *scales.ticks.height*
 \$("#container").ejLinearGauge({
 scales: [{
 ticks: [{ type: "minorinterval", height: 8 }]
 }]
 }]);
Property: *axes.minorTicks.height*
 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 minorTicks: { height : 10 }
 }];
 gauge.appendTo('#container');

| Width of Minor Ticks | **Property:** *scales.ticks.width*
 \$("#container").ejLinearGauge({
 scales:[
 ticks: [{ type: "minorinterval", width: 5 }]
 }]
 }]);
Property: *axes.minorTicks.width*
 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 minorTicks: { width : 2 }
 }];
 gauge.appendTo('#container');

| Color of Minor Ticks | **Property:** *scales.ticks.color*
 \$("#container").ejLinearGauge({
 scales:[
 ticks: [{ type: "minorinterval", color: "Blue" }]
 }]);
Property: *axes.minorTicks.color*
 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 minorTicks: { color: "Blue: }
 }];
 gauge.appendTo('#container');

| Offset for Minor Ticks | **Property:** *scales.ticks.distanceFromScale*
 \$("#container").ejLinearGauge({
 scales:[
 ticks: [{ distanceFromScale:
 { x: 5, y: 5 }, type: "minorinterval" }]
 }]);
Property: *axes.minorTicks.offset*
 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 minorTicks: { offset
 : 1 }
 }];
 gauge.appendTo('#container');

| Interval of Minor Ticks | **Property:** *scales.minorIntervalValue*
 \$("#container").ejLinearGauge({
 scales: [{ minorIntervalValue: 8 }]
 }]);
Property:

axes.minorTicks.interval
let gauge: LinearGauge = new LinearGauge({ axes: [{
minorTicks: { interval : 5 } }]; gauge.appendTo('#container');

| Angle of Minor Ticks | **Property:** *scales.ticks.angle* \$("#container").ejLinearGauge({ scales: [{ ticks: [{ type: "minorinterval", angle: 30 }] }]; | Not Applicable |

| Opacity of Minor Ticks | **Property:** *scales.ticks.opacity* \$("#container").ejLinearGauge({ scales: [{ ticks: [{ type: "minorinterval", opacity: 0.5 }] }]; | Not Applicable |

Labels

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Angle | **Property:** *scales.labels.angle* \$("#container").ejLinearGauge({ scales: [{ labels: [{ angle: 15 }] }]; | Not Applicable |

| Offset | **Property:** *scales.labels.distanceFromScale* \$("#container").ejLinearGauge({ scales: [{ labels: [{ distanceFromScale: { x: -5, y: 10 } }] }]; **Property:** *axes.labelStyle.offset* let gauge: LinearGauge = new LinearGauge({ axes: [{ labelStyle: { offset : 3 } }]; gauge.appendTo('#container');

| Format | **Property:** *scales.labels.unitText* \$("#container").ejLinearGauge({ scales: [{ labels: [{ unitText: "F" }] }]; **Property:** *axes.labelStyle.format* let gauge: LinearGauge = new LinearGauge({ axes: [{ labelStyle: { format: 'c' } }]; gauge.appendTo('#container');

| Unit Text Placement | **Property:** *scales.labels.unitTextPlacement* \$("#container").ejLinearGauge({ scales: [{ labels: [{ unitTextPlacement: "front" }] }]; | Not Applicable |

| Label Range Color | Not Applicable | **Property:** *axes.labelStyle.useRangeColor* let gauge: LinearGauge = new LinearGauge({ axes: [{ labelStyle: { useRangeColor: true } }]; gauge.appendTo('#container');

| Opacity | **Property:** *scales.labels.opacity* \$("#container").ejLinearGauge({ scales: [{ labels: [{ opacity: 0.5 }] }]; **Property:** *axes.labelStyle.font.opacity* let gauge: LinearGauge = new LinearGauge({ axes: [{ labelStyle: { font: { opacity: 5 } } }]; gauge.appendTo('#container');

| Label Text Color | **Property:** *scales.labels.textColor* \$("#container").ejLinearGauge({ scales: [{ labels: [{ textColor: "Red" }] }]; **Property:** *axes.labelStyle.font.color* let gauge: LinearGauge = new LinearGauge({ axes: [{ labelStyle: { font: { color: 'red' } } }]; gauge.appendTo('#container');

| Label Font Family | **Property:** *scales.labels.font.fontFamily* \$("#container").ejLinearGauge({ scales: [{ labels: [{ font: { fontFamily: "Segoe UI" } }] }]; **Property:** *axes.labelStyle.font.fontFamily* let gauge: LinearGauge = new LinearGauge({

```
</> &#160; axes: [{ </> &#160; &#160; labelStyle: { font:{ fontFamily: 'Arial' } } </> &#160; } ] };
</> gauge.appendTo('#container');
```

| Label Font Style | **Property:** *scales.labels.font.fontStyle*

 \$("#container").ejLinearGauge({

 scales:[{
 labels: [{ font: {
 fontStyle:
 ej.datavisualization.LinearGauge.FontStyle.Normal } }]
 }] }; | **Property:**
axes.labelStyle.font.fontStyle

 let gauge: LinearGauge = new LinearGauge({
 axes:
 [{
 labelStyle: { font: { fontStyle: 'Bold' } }
 }] };

 gauge.appendTo('#container');

| Label Size | **Property:** *scales.labels.font.size*

 \$("#container").ejLinearGauge({

 scales:[{
 labels: [{ font: { size: "20px" } }]
 }] }; | **Property:**
axes.labelStyle.font.size

 let gauge: LinearGauge = new LinearGauge({
 axes: [{

 labelStyle: { font:{ size: "15px" } }
 }] };

 gauge.appendTo('#container');

| Label Font Weight | Not Applicable | **Property:** *axes.labelStyle.font.fontWeight*

 let gauge:
 LinearGauge = new LinearGauge({
 axes: [{
 labelStyle: { font:{
 fontWeight: '4' } }
 }] };
 gauge.appendTo('#container');

Axis

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Minimum Value | **Property:** *scales.minimum*

 \$("#container").ejLinearGauge({

 scales: [{ minimum: 20 }]
 }]; | **Property:** *axes.minimum*

 let gauge: LinearGauge = new
 LinearGauge({
 axes: [{ minimum: 20 }]
 }];
 gauge.appendTo('#container');

| Maximum Value | **Property:** *scales.maximum*

 \$("#container").ejLinearGauge({

 scales: [{ maximum: 120 }]
 }]; | **Property:** *axes.maximum*

 let gauge: LinearGauge = new
 LinearGauge({
 axes: [{ maximum: 120 }]
 }];
 gauge.appendTo('#container');

| Inverted Position | Not Applicable | **Property:** *axes.isInversed*

 let gauge: LinearGauge = new
 LinearGauge({
 axes: [{ isInversed: true }]
 }];
 gauge.appendTo('#container');

| Opposed Position | Not Applicable | **Property:** *axes.opposedPosition*

 let gauge: LinearGauge
 = new LinearGauge({
 axes: [{ opposedPosition: true }]
 }];

 gauge.appendTo('#container');

Ranges

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Start Value | **Property:** *scales.ranges.startValue*

 \$("#container").ejLinearGauge({

 scales:[{
 ranges: [{ startValue: 20 }]
 }] }; | **Property:**
axes.ranges.start

 let gauge: LinearGauge = new LinearGauge({
 axes: [{

 ranges: [{ start: 20 }]
 }] };
 gauge.appendTo('#container');

| End Value | **Property:** *scales.ranges.endValue*

 \$("#container").ejLinearGauge({

 scales:[{
 ranges: [{ endValue: 20 }]
 }] }; | **Property:**
axes.ranges.end

 let gauge: LinearGauge = new LinearGauge({
 axes: [{

 ranges: [{ end: 20 }]
 }] };
 gauge.appendTo('#container');

|Start Width| **Property:** *scales.ranges.startWidth*
 let gauge: LinearGauge = new LinearGauge({ scales: [{ ranges: [{ startWidth: 10 }] }] });
 gauge.appendTo('#container');

|End Width| **Property:** *scales.ranges.endWidth*
 let gauge: LinearGauge = new LinearGauge({ scales: [{ ranges: [{ endWidth: 15 }] }] });
 gauge.appendTo('#container');

|Color| **Property:** *scales.ranges.backgroundColor*
 let gauge: LinearGauge = new LinearGauge({ scales: [{ ranges: [{ backgroundColor: "red" }] }] });
 gauge.appendTo('#container');

|Offset| **Property:** *scales.ranges.distanceFromScale*
 let gauge: LinearGauge = new LinearGauge({ scales: [{ ranges: [{ distanceFromScale: 10 }] }] });
 gauge.appendTo('#container');

|Range Position| **Property:** *scales.ranges.placement*
 let gauge: LinearGauge = new LinearGauge({ scales: [{ ranges: [{ placement: "Near" }] }] });
 gauge.appendTo('#container');

|Opacity| **Property:** *scales.ranges.opacity*
 scales: [{ ranges: [{ opacity: 0.3 }] }];
 Not Applicable

|Border Customization| **Property:** *scales.ranges.border*
 let gauge: LinearGauge = new LinearGauge({ scales: [{ ranges: [{ border: { color: "green", width: 2 } }] }] });
 let gauge: LinearGauge = new LinearGauge({ axes: [{ ranges: [{ border: { color: "blue", width: 2 } }] }] });
 gauge.appendTo('#container');

Bar Pointer

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

|Bar Pointer| **Property:** *scales.barPointers.value*
 let gauge: LinearGauge = new LinearGauge({ scales: [{ barPointers: [{ value: 20 }] }] });
 let gauge: LinearGauge = new LinearGauge({ axes: [{ pointers: [{ type: 'Bar', value: 20 }] }] });
 gauge.appendTo('#container');

|Color of Bar Pointer| **Property:** *scales.barPointers.backgroundColor*
 let gauge: LinearGauge = new LinearGauge({ scales: [{ barPointers: [{ value: 20, backgroundColor: "red" }] }] });
 let gauge: LinearGauge = new LinearGauge({ axes: [{ pointers: [{ type: 'Bar', value: 20, color: 'red' }] }] });
 gauge.appendTo('#container');

| Offset of Bar Pointer | **Property:** *scales.barPointers.distanceFromScale*
 \$("#container").ejLinearGauge({ scales: [{ value: 40, distanceFromScale: 20 }]; | **Property:** *axes.pointers.offset* let gauge: LinearGauge = new LinearGauge({ axes: [{ pointers: [{ type: 'Bar', value: 20, offset: 20 }]; gauge.appendTo('#container'); |

| Opacity of Bar Pointer | **Property:** *scales.barPointers.opacity*
 \$("#container").ejLinearGauge({ scales: [{ value: 40, opacity: 0.5 }]; | **Property:** *axes.pointers.opacity* let gauge: LinearGauge = new LinearGauge({ axes: [{ pointers: [{ type: 'Bar', value: 20, opacity: 0.5 }]; gauge.appendTo('#container'); |

| Width of Bar Pointer | **Property:** *scales.barPointers.width*
 \$("#container").ejLinearGauge({ scales: [{ value: 40, width: 25 }]; | **Property:** *axes.pointers.width* let gauge: LinearGauge = new LinearGauge({ axes: [{ pointers: [{ type: 'Bar', value: 20, width: 25 }]; gauge.appendTo('#container'); |

| Gradients of Bar Pointer | **Property:** *scales.barPointers.gradients*
 \$("#container").ejLinearGauge({ scales: [{ value: 40, gradients: { colorInfo: [{ colorStop : 0, color: "#FFFFFF" }] }]; | Not Applicable |

| Border of Bar Pointer | **Property:** *scales.barPointers.border*
 \$("#container").ejLinearGauge({ scales: [{ value: 40, border: { color: "red", width: 2 } }]; | **Property:** *axes.pointers.border* let gauge: LinearGauge = new LinearGauge({ axes: [{ pointers: [{ type: 'Bar', value: 20, border: { color: 'red', width: 2.5 } }]; gauge.appendTo('#container'); |

| Animation of Bar Pointer | **Property:** *enableAnimation*
 \$("#container").ejLinearGauge({ enableAnimation: true }; | **Property:** *axes.pointers.animationDuration* let gauge: LinearGauge = new LinearGauge({ axes: [{ pointers: [{ type: 'Bar', value: 20, animationDuration: 2500 }]; gauge.appendTo('#container'); |

| Rounded Corner of Bar Pointer | Not Applicable | **Property:** *axes.pointers.roundedCornerRadius* let gauge: LinearGauge = new LinearGauge({ axes: [{ pointers: [{ type: 'Bar', value: 20, roundedCornerRadius: 15 }]; gauge.appendTo('#container'); |

Marker Pointer

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Marker Pointer | **Property:** *scales.markerPointers.value*
 \$("#container").ejLinearGauge({ scales: [{ markerPointers: [{ value: 20 }]; | **Property:** *axes.pointers.value* let gauge: LinearGauge = new LinearGauge({ axes: [{


```
&#160; &#160; pointers: [{ type: 'Marker', value: 20 } <br/> &#160; } &#160; }]; <br/>
gauge.appendTo('#container');|
```

| Color of Marker Pointer | **Property:** *scales.markerPointers.backgroundColor*

 \$("#container").ejLinearGauge({
 scales:[{
 markerPointers: [{ value: 20, backgroundColor: "blue" }]
 } }]; | **Property:** *axes.pointers.color*

 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 pointers: [{ type: 'Marker', value: 20, color: 'red' }]
 } }];
 gauge.appendTo('#container');|

| Offset of Marker Pointer | **Property:** *scales.markerPointers.distanceFromScale*

 \$("#container").ejLinearGauge({
 scales:[{
 markerPointers: [{ value: 40, distanceFromScale: 10 }]
 } }]; | **Property:** *axes.pointers.offset*

 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 pointers: [{ type: 'Marker', value: 20, offset: 10 }]
 } }];
 gauge.appendTo('#container');|

| Opacity of Marker Pointer | **Property:** *scales.markerPointers.opacity*

 \$("#container").ejLinearGauge({
 scales:[{
 markerPointers: [{ value: 40, opacity: 1 }]
 } }]; | **Property:** *axes.pointers.opacity*

 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 pointers: [{ type: 'Marker', value: 20, opacity: 1 }]
 } }];
 gauge.appendTo('#container');|

| Width of Marker Pointer | **Property:** *scales.markerPointers.width*

 \$("#container").ejLinearGauge({
 scales:[{
 markerPointers: [{ value: 40, width: 20 }]
 } }]; | **Property:** *axes.pointers.width*

 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 pointers: [{ type: 'Marker', value: 20, width: 25 }]
 } }];
 gauge.appendTo('#container');|

| Gradients of Marker Pointer | **Property:** *scales.markerPointers.gradients*

 \$("#container").ejLinearGauge({
 scales:[{
 markerPointers: [{
 value: 40,
 gradients:{colorInfo:[{colorStop : 0, color:"#FFFFFF"}]}
 } } }]; | Not Applicable|

| Border of Marker Pointer | **Property:** *scales.markerPointers.border*

 \$("#container").ejLinearGauge({
 scales:[{
 markerPointers: [{
 value: 40,
 border: { color: "red", width: 2 } }]
 } }]; | **Property:** *axes.pointers.border*

 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 pointers: [{
 type: 'Marker',
 value: 20,
 border: { color: 'red', width: 2.5 } }]
 } }];
 gauge.appendTo('#container');|

| Animation of Marker Pointer | **Property:** *enableMarkerPointerAnimation*

 \$("#container").ejLinearGauge({
 enableMarkerPointerAnimation: true
 } }]; | **Property:** *axes.pointers.animationDuration*

 let gauge: LinearGauge = new LinearGauge({
 axes: [{
 pointers: [{
 type: 'Marker',
 value: 20,
 animationDuration: 1000 }]
 } }];
 gauge.appendTo('#container');|

| Type of Marker Pointer | **Property:** *scales.markerPointers.type*

 \$("#container").ejLinearGauge({
 scales:[{
 markerPointers: [{
 value: 40,
 type: ej.datavisualization.LinearGauge.
 MarkerType.Diamond }]
 } }]; | **Property:** *axes.pointers.markerType*

 let gauge: LinearGauge = new LinearGauge({
 axes: [{

```
<br/> &#160; &#160; pointers: [{ <br/> &#160; &#160; &#160; type: 'Marker', <br/> &#160; &#160; &#160; &#160; value: 20, <br/> &#160; &#160; &#160; &#160; markerType: 'Diamond' }] <br/> &#160; &#160; &#160; &#160; gauge.appendTo('#container');|
```

```
| Placement of Marker Pointer | Property: scales.markerPointers.placement  
$("#container").ejLinearGauge({ scales: [{ markerPointers: [{  
&#160; &#160; &#160; value: 40, &#160; &#160; &#160; placement:  
ej.datavisualization.LinearGauge. &#160; &#160; &#160; &#160; PointerPlacement.Near }} &#160; &#160; &#160; &#160; ] }]); Property: axes.pointers.placement  
let gauge: LinearGauge = new LinearGauge({  
&#160; &#160; axes: [{ pointers: [{ type: 'Marker',  
&#160; &#160; &#160; value: 20, &#160; &#160; &#160; placement: 'Center' }]  
&#160; &#160; &#160; &#160; } ]}); gauge.appendTo('#container');
```

```
| [Drag of Marker Pointer] Property: readOnly  
false } | Property: axes.pointers.enableDrag let gauge: LinearGauge = new LinearGauge({  
  axes: [{  
    pointers: [{  
      type: 'Marker',  
      value: 20,  
      enableDrag: true }]  
    }  
  ]  
}); gauge.appendTo('#container');
```

```
| Image Marker Pointer| Not Applicable| Property: axes.pointers.imageUrl<br/><br/> let gauge:
LinearGauge = new LinearGauge({ <br/> &#160; axes: [{ <br/> &#160; &#160; pointers: [{ <br/> &#160;
&#160; &#160; type: 'Marker', <br/> &#160; &#160; &#160; value: 20, <br/> &#160; &#160; &#160;
imageUrl: './image.png' }] <br/> &#160; }]); <br/> gauge.appendTo('#container');
```

Annotation

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

— — — — —

```
| Content| Property: scales.customLabels.value<br/><br/> $(("#container").ejLinearGauge({ <br/>
&#160; scales: [{ <br/> &#160; &#160; showCustomLabels: true, <br/> &#160; &#160; customLabels: [{
<br/> &#160; &#160; &#160; value: "LinearGauge" } ] <br/> &#160; }]);| Property:
annotations.content<br/><br/> let gauge: LinearGauge = new LinearGauge({ <br/> &#160; annotation: [{
<br/> &#160; &#160; &#160; content: "Annotation" } ] }); <br/> gauge.appendTo('#container');|
```

```
|Horizontal Alignment| Not Applicable| Property: annotations.horizontalAlignment<br/><br/> let
gauge: LinearGauge = new LinearGauge({ <br/> &#160; annotation: [{ <br/> &#160; &#160; content:
"Annotation", horizontalAlignment: 'Center' } ]}); <br/> gauge.appendTo('#container');
```

```
|Vertical Alignment| Not Applicable| Property: annotations.verticalAlignment<br/><br/> let gauge:
LinearGauge = new LinearGauge({ <br/> &#160; annotation: [{ <br/> &#160; &#160; content:
"Annotation", verticalAlignment: 'Far' } ] }; <br/> gauge.appendTo('#container');
```

```
| Position of X| Property: scales.customLabels.position.x<br/><br/> $(" #container").ejLinearGauge({  
<br/> &#160; scales: [{ <br/> &#160; &#160; showCustomLabels: true, <br/> &#160; &#160;  
customLabels: [ { <br/> &#160; &#160; &#160; value: "LinearGauge", position: { x: 20 } } ] <br/> &#160; &#160; } ]};  
Property: annotations.x<br/><br/> let gauge: LinearGauge = new LinearGauge({ <br/> &#160;  
annotation: [ { <br/> &#160; &#160; content: "Annotation", x: 35 } ] }); <br/>  
gauge.appendTo(' #container'); |
```

```
| Position of Y | Property: scales.customLabels.position.y<br/><br/> $(" #container").ejLinearGauge({<br/> &#160; scales: [{<br/> &#160; &#160; showCustomLabels: true, <br/> &#160; &#160; customLabels: [{<br/> &#160; &#160; &#160; value: "LinearGauge", position: { y: 30 } }]}<br/> &#160; }
```

}); | **Property:** *annotations.y*
 let gauge: LinearGauge = new LinearGauge({
 annotation: [{ content: "Annotation", y: 40 }] };
 gauge.appendTo('#container');

| Z Index | Not Applicable | **Property:** *annotations.zIndex*
 let gauge: LinearGauge = new
 LinearGauge({
 annotation: [{ content: "Annotation", zIndex: 1 }] };
 gauge.appendTo('#container');

| Axis Index | Not Applicable | **Property:** *annotations.axisIndex*
 let gauge: LinearGauge = new
 LinearGauge({
 annotation: [{ content: "Annotation", axisIndex: 0 }]
 });
 gauge.appendTo('#container');

| Axis Value | Not Applicable | **Property:** *annotations.axisValue*
 let gauge: LinearGauge = new
 LinearGauge({
 annotation: [{ content: "Annotation", axisValue: 35 }]
 });
 gauge.appendTo('#container');

| Font customization | **Property:** *scales.customLabels.font*
 \$("#container").ejLinearGauge({
 scales: [{
 showCustomLabels: true,
 customLabels: [{ value: "LinearGauge", font: { size: "30px" } }]
 }]; | **Property:** *annotations.font*
 let gauge: LinearGauge = new LinearGauge({
 annotation: [{ content: "Annotation", font: { size: '15px' } }] };
 gauge.appendTo('#container');

| Annotation Color | **Property:** *scales.customLabels.color*
 \$("#container").ejLinearGauge({
 scales: [{
 showCustomLabels: true,
 customLabels: [{ value: "LinearGauge", color: "yellow" }]
 }]; | **Property:** *annotations.font*
 let gauge: LinearGauge = new LinearGauge({
 annotation: [{ content: "Annotation", font: { color: 'red' } }] };
 gauge.appendTo('#container');

| Opacity of Annotation | **Property:** *scales.customLabels.opacity*
 \$("#container").ejLinearGauge({
 scales: [{
 showCustomLabels: true,
 customLabels: [{ value: "LinearGauge", opacity: 0.5 }]
 }]; | **Property:** *annotations.font*
 let gauge: LinearGauge = new LinearGauge({
 annotation: [{ content: "Annotation", font: { opacity: 0.7 } }] };
 gauge.appendTo('#container');

| Position Type | **Property:** *scales.customLabels.positionType*
 \$("#container").ejLinearGauge({
 scales: [{
 showCustomLabels: true,
 customLabels: [{ value: "LinearGauge", positionType: "outer" }]
 }]; | Not applicable

| TextAngle of Annotation | **Property:** *scales.customLabels.textAngle*
 \$("#container").ejLinearGauge({
 scales: [{
 showCustomLabels: true,
 customLabels: [{ value: "LinearGauge", textAngle: 25
 }]
 }]; | Not applicable

Tooltip

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Tooltip for Pointer| Not Applicable| **Property:** *tooltip.enable*
 let gauge: LinearGauge = new LinearGauge({ tooltip: { enable: true } });
 gauge.appendTo('#container');

| Tooltip for Label| **Property:** *tooltip.showLabelTooltip*
 \$("#container").ejLinearGauge({ tooltip: { showLabelTooltip: true } });
 | Not Applicable|

| Tooltip Format| Not Applicable| **Property:** *tooltip.format*
 let gauge: LinearGauge = new LinearGauge({ tooltip: { enable: true, format: \${pointers.value}cm } });
 gauge.appendTo('#container');

| Tooltip Color| Not Applicable| **Property:** *tooltip.fill*
 let gauge: LinearGauge = new LinearGauge({ tooltip: { enable: true, fill: 'gray' } });
 gauge.appendTo('#container');

| Tooltip Template| **Property:** *tooltip.templateID*
 \$("#container").ejLinearGauge({ tooltip: { templateID: true } });
Property: *tooltip.template*
 let gauge: LinearGauge = new LinearGauge({ tooltip: { enable: true, template: 'Template' } });
 gauge.appendTo('#container');

| Tooltip Animation| Not Applicable| **Property:** *tooltip.enableAnimation*
 let gauge: LinearGauge = new LinearGauge({ tooltip: { enable: true, enableAnimation: true } });
 gauge.appendTo('#container');

| Tooltip Border| Not Applicable| **Property:** *tooltip.border*
 let gauge: LinearGauge = new LinearGauge({ tooltip: { enable: true, border: { width: 2, color: 'red' } } });
 gauge.appendTo('#container');

| Tooltip TextStyle| Not Applicable| **Property:** *tooltip.textStyle*
 let gauge: LinearGauge = new LinearGauge({ tooltip: { enable: true, textStyle: { color: 'white', size: '10px' } } });
 gauge.appendTo('#container');

Appearance of Linear Gauge

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Background Color| **Property:** *backgroundColor*
 \$("#container").ejLinearGauge({ backgroundColor: "Red" });
Property: *background*
 let gauge: LinearGauge = new LinearGauge({ background: 'skyblue' });
 gauge.appendTo('#container');

| Border Color| **Property:** *borderColor*
 \$("#container").ejLinearGauge({ borderColor: "Black" });
Property: *border.color*
 let gauge: LinearGauge = new LinearGauge({ border: { color: 'red' } });
 gauge.appendTo('#container');

| Margin| Not Applicable| **Property:** *margin*
 let gauge: LinearGauge = new LinearGauge({ margin: { left: 40, right: 40, top: 40, bottom: 40 } });
 gauge.appendTo('#container');

| Orientation| **Property:** *orientation*
 \$("#container").ejLinearGauge({ orientation: "Vertical" });
Property: *orientation*
 let gauge: LinearGauge = new LinearGauge({ orientation: 'Horizontal' });
 gauge.appendTo('#container');

| Locale| **Property:** *locale*
 \$("#container").ejLinearGauge({ locale: "en-US" });| **Property:** *locale*
 let gauge: LinearGauge = new LinearGauge({ locale: 'en-US' }
); gauge.appendTo('#container');

| Theme| **Property:** *theme*
 \$("#container").ejLinearGauge({ theme: ej.datavisualization.LinearGauge.flatdark });| **Property:** *theme*
 let gauge: LinearGauge = new LinearGauge({ theme: 'Highcontrast' }
); gauge.appendTo('#container');

| Gauge Title| Not Applicable| **Property:** *title*
 let gauge: LinearGauge = new LinearGauge({ title: 'Linear Gauge' }
); gauge.appendTo('#container');

Gauge Container type

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Container Type| **Property:** *scales.type*
 \$("#container").ejLinearGauge({ scales: { type: 'thermometer' } }
);| **Property:** *container.type*
 let gauge: LinearGauge = new LinearGauge({ container: { type: 'Thermometer' } }
); gauge.appendTo('#container');

| Container Height| Not Applicable| **Property:** *container.height*
 let gauge: LinearGauge = new LinearGauge({ container: { height: 20 } }
); gauge.appendTo('#container');

| Container Width| Not Applicable| **Property:** *container.width*
 let gauge: LinearGauge = new LinearGauge({ container: { width: 10 } }
); gauge.appendTo('#container');

| Container Offset| Not Applicable| **Property:** *container.offset*
 let gauge: LinearGauge = new LinearGauge({ container: { offset: 5 } }
); gauge.appendTo('#container');

Events

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Animation Complete Event| Not Applicable| **Event:** *animationComplete*
 let gauge: LinearGauge = new LinearGauge({ animationComplete: function(e: IAnimationCompleteEventArgs): void { } }
); gauge.appendTo('#container');

| Annotation Render Event| **Event:** *drawCustomLabel*
 \$("#container").ejLinearGauge({ drawCustomLabel: function (args) { } }
);| **Event:** *annotationRender*
 let gauge: LinearGauge = new LinearGauge({ annotationRender: function(e: IAnnotationRenderEventArgs): void { } }
); gauge.appendTo('#container');

| AxisLabel Render Event| **Event:** *drawLabels*
 \$("#container").ejLinearGauge({ drawLabels: function (args) { } }
);| **Event:** *axisLabelRender*
 let gauge: LinearGauge = new LinearGauge({ axisLabelRender: function(e: IAxisLabelRenderEventArgs): void { } }
); gauge.appendTo('#container');

| Load Event| **Event:** *load*
 \$("#container").ejLinearGauge({ load: function (args) {}

});| **Event:** *load*
 let gauge: LinearGauge = new LinearGauge({ load:
 function(e: ILoadEventArgs): void { }
 }
 gauge.appendTo('#container');|

| Loaded Event| Not Applicable| **Event:** *loaded*
 let gauge: LinearGauge = new LinearGauge({
 loaded: function(e: ILoadedEventArgs): void { }
 }

 gauge.appendTo('#container');|

| Resize Event| Not Applicable| **Event:** *resized*
 let gauge: LinearGauge = new LinearGauge({
 resized: function(e: IResizeEventArgs): void { }
 }

 gauge.appendTo('#container');|

| Tooltip Render Event| Not Applicable| **Event:** *tooltipRender*
 let gauge: LinearGauge = new
 LinearGauge({ tooltipRender: function(e: ITooltipRenderEventArgs): void { }
 }

 gauge.appendTo('#container');|

| Value Change Event| Not Applicable| **Event:** *valueChange*
 let gauge: LinearGauge = new
 LinearGauge({ valueChange: function(e: IValueChangeEventArgs): void { }
 }

 gauge.appendTo('#container');|

| Mouse Move Event| **Event:** *mouseClickMove*
 \$("#container").ejLinearGauge({
 mouseClickMove: function (args) {}
 });| **Event:** *gaugeMouseMove*
 let gauge:
 LinearGauge = new LinearGauge({ gaugeMouseMove: function(e: IMouseEventArgs): void
 { }
 }
 gauge.appendTo('#container');|

| Mouse Up Event| **Event:** *mouseClickUp*
 \$("#container").ejLinearGauge({
 mouseClickUp: function (args) {}
 });| **Event:** *gaugeMouseUp*
 let gauge: LinearGauge =
 new LinearGauge({ gaugeMouseUp: function(e: IMouseEventArgs): void { }
 }

 gauge.appendTo('#container');|

| Mouse Down Event| Not Applicable| **Event:** *gaugeMouseDown*
 let gauge: LinearGauge =
 new LinearGauge({ gaugeMouseDown: function(e: IMouseEventArgs): void { }
 }

 gauge.appendTo('#container');|

| Mouse Leave Event| Not Applicable| **Event:** *gaugeMouseLeave*
 let gauge: LinearGauge =
 new LinearGauge({ gaugeMouseLeave: function(e: IMouseEventArgs): void { }
 }

 gauge.appendTo('#container');|

| Mouse Click Event| **Event:** *mouseClick*
 \$("#container").ejLinearGauge({
 mouseClick: function (args) {}
 });| Not Applicable|

| Render Complete Event| **Event:** *renderComplete*
 \$("#container").ejLinearGauge({ renderComplete: function (args) {}
 });| Not
 Applicable|

| Double Click Event| **Event:** *doubleClick*
 \$("#container").ejLinearGauge({
 doubleClick: function (args) {}
 });| Not Applicable|

| Right Click Event| **Event:** *rightClick*
 \$("#container").ejLinearGauge({ rightClick:
 function (args) {}
 });| Not Applicable|

| BarPointers Event| **Event:** *drawBarPointers*
 \$("#container").ejLinearGauge({
 drawBarPointers: function (args) {}
 });| Not Applicable|

| Indicators Event| **Event:** *drawIndicators*
 \$("#container").ejLinearGauge({
 drawIndicators: function (args) {}
 });| Not Applicable|

| MarkerPointer Event| **Event:** *drawMarkerPointers*

 \$("#container").ejLinearGauge({
 drawMarkerPointers: function (args) {}
 });| Not Applicable|

| Ranges Event| **Event:** *drawRange*

 \$("#container").ejLinearGauge({
 drawRange: function (args) {}
 });| Not Applicable|

| Gauge Initialized Event| **Event:** *init*

 \$("#container").ejLinearGauge({
 init: function (args) {}
 });| Not Applicable|

List Box

Getting started in EJ2 JavaScript List box control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

Component Initialization

The Essential JS 2 JavaScript components can be initialized by using either of the following ways.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

Using local script and style references in a HTML page

Step 1: Create an app folder **myapp** for Essential JS 2 JavaScript components.

Step 2: You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

Syntax:

Script: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js**

Styles: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css**

Example:

Script: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-dropdowns/dist/global/ej2-dropdowns.min.js**

Styles: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-dropdowns/styles/material.css**

Step 3: Create a folder **myapp/resources** and copy/paste the global scripts and styles from the above installed location to **myapp/resources** location.

Step 4: Create a HTML page (index.html) in **myapp** location and add the Essentials JS 2 script and style references.

`html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 ListBox's global script -->
<script src="resources/ej2-dropdowns.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Step 5: Now, add the `input` element and initiate the `Essential JS 2 ListBox` component in the `index.html` by using following code

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 ListBox's global script -->
<script src="resources/ej2-dropdowns.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <input> element -->
<input id='listbox' />
<script>
// initialize ListBox component
var listObj = new ej.dropdowns.ListBox();
// Render initialized ListBox.
listObj.appendTo('#listbox');
```



```

</script>
</body>
</html>
`

```

Step 6: Now, run the `index.html` in web browser, it will render the **Essential JS 2 ListBox** component.

Using CDN link for script and style reference

Step 1: Create an app folder `myapp` for the Essential JS 2 JavaScript components.

Step 2: The Essential JS 2 component's global scripts and styles are already hosted in the below CDN link formats.

Syntax:

Script: `http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Styles: `http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

Example:

Script: <http://cdn.syncfusion.com/ej2/ej2-dropdowns/dist/global/ej2-dropdowns.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css>

Step 3: Create a HTML page (`index.html`) in `myapp` location and add the CDN link references. Now, add the `input` element and initiate the **Essential JS 2 ListBox** component in the `index.html` by using following code.

INDEX.HTML

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>

    <title>Essential JS 2</title>
    <!-- Essential JS 2 material theme -->
    <link href="http://cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="http://cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="http://cdn.syncfusion.com/ej2/21.2.3/ej2-
dropdowns/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="http://cdn.syncfusion.com/ej2/21.2.3/ej2-
buttons/styles/material.css" rel="stylesheet" type="text/css"/>
    <!-- Essential JS 2 ListBox's global script -->
    <script src="http://cdn.syncfusion.com/ej2/21.2.3/ej2-
dropdowns/dist/global/ej2-dropdowns.min.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
  </head>
  <body>
    <!-- Add the HTML <input> element -->

```

```
<input id='listbox' />
<script>
  // initialize ListBox component
  var listObj = new ej.dropdowns.ListBox();
  // Render initialized ListBox.
  listObj.appendTo('#listbox');
</script>
</body>
</html>
```

Step 4: Now, run the `index.html` in web browser, it will render the `Essential JS 2 ListBox` component.

Binding data source

After initialization, populate the ListBox with data using the `dataSource` property. Here, an array of object values is passed to the ListBox component.

``html`

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 ListBox's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-dropdowns/dist/global/ej2-dropdowns.min.js"
type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <input> element -->
<input id='listbox' />
<script>
var data = [
{ text: 'Hennessey Venom', id: 'list-01' },
{ text: 'Bugatti Chiron', id: 'list-02' },
```

```

{ text: 'Bugatti Veyron Super Sport', id: 'list-03' },
{ text: 'SSC Ultimate Aero', id: 'list-04' },
{ text: 'Koenigsegg CCR', id: 'list-05' },
{ text: 'McLaren F1', id: 'list-06' },
{ text: 'Aston Martin One- 77', id: 'list-07' },
{ text: 'Jaguar XJ220', id: 'list-08' },
{ text: 'McLaren P1', id: 'list-09' },
{ text: 'Ferrari LaFerrari', id: 'list-10' },
];
// initialize ListBox component
var listObj = new ej.dropdowns.ListBox({
dataSource: data
});
listObj.appendTo('#listbox');
</script>
</body>
</html>
`

```

See Also

- [How to bind data to the list box](#)
- [How to initialize list box using different tags](#)

Data binding in EJ2 JavaScript List box control

The ListBox loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of `array` or `DataManager`.

The ListBox also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON, and JSONP with the help of `DataManager` adaptors.

Fields	Type	Description
<code>text</code>	<code>string</code>	Specifies the display text of each list item.
<code>value</code>	<code>string</code>	Specifies the hidden data value mapped to each list item that should contain a unique value.
<code>groupBy</code>	<code>string</code>	Specifies the category under which the list item has to be grouped.
<code>iconCss</code>	<code>string</code>	Specifies the iconCss class that needs to be mapped.

| [htmlAttributes](#) | `string` | Allows additional attributes to configure the elements in various ways to meet the criteria. |

When binding complex data to the ListBox, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Local Data

Local data can be represented by the following ways as described below.

Array of string

The ListBox has support to load array of primitive data such as strings or numbers. Here, both value and text field acts as same.

INDEX.TS

```
import { ListBox } from '@syncfusion/ej2-dropdowns';
// define array of string
let sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis', 'Basket Ball', 'Base Ball', 'Hockey', 'Volley Ball'];
// initialize ListBox component
let listObj: ListBox = new ListBox({
    //set the data to dataSource property
    dataSource: sportsData
});
listObj.appendTo('#listbox');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <h4>Select your favorite sports:</h4>
    <input id="listbox">
  </div>
```

```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Array of object

The ListBox can generate its list items through an array of object data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **id** and **sports** column from complex data have been mapped to the **value** field and **text** field, respectively.

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
let sportsData: { [key: string]: Object }[] = [
    { id: 'game1', sports: 'Badminton' },
    { id: 'game2', sports: 'Cricket' },
    { id: 'game3', sports: 'Football' },
    { id: 'game4', sports: 'Golf' },
    { id: 'game5', sports: 'Tennis' },
    { id: 'game6', sports: 'Basket Ball' },
    { id: 'game7', sports: 'Base Ball' },
    { id: 'game8', sports: 'Hockey' },
    { id: 'game9', sports: 'Volley Ball' }];
// initialize ListBox component
let listObj: ListBox = new ListBox({
    //set the dataSource property
    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { text: 'sports', value: 'id' }
});
listObj.appendTo('#listbox');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 ListBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <h4>Select your favorite sports:</h4>
        <input id="listbox">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Array of complex object

The ListBox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Sports.Name** column from complex data have been mapped to the **text** field.

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
let sportsData: { [key: string]: Object }[] = [
    { Id: 'game1', Sports: { Name: 'Badminton' } },
    { Id: 'game2', Sports: { Name: 'Cricket' } },
    { Id: 'game3', Sports: { Name: 'Football' } },
    { Id: 'game4', Sports: { Name: 'Golf' } },
    { Id: 'game5', Sports: { Name: 'Tennis' } },
    { Id: 'game6', Sports: { Name: 'Basket Ball' } },
    { Id: 'game7', Sports: { Name: 'Base Ball' } },
    { Id: 'game8', Sports: { Name: 'Hockey' } },
    { Id: 'game9', Sports: { Name: 'Volley Ball' } }];
// initialize ListBox component
let listObj: ListBox = new ListBox({
    //set the dataSource property
    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { text: 'Sports.Name' }
});
listObj.appendTo('#listbox');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 ListBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <h4>Select your favorite sports:</h4>
        <input id="listbox">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The ListBox has also be rendered through `<input>`, ``, `select` element.

Remote Data

The ListBox supports retrieval of data from remote data services with the help of [DataManager](#) component. The [Query](#) property is used to fetch data from the database and bind it to the ListBox.

The following sample displays the first 10 products from `Products` table of the `Northwind` Data Service.

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let listObj: ListBox = new ListBox({
    //bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor
    }),
    //bind the Query instance to query property
    query: new
Query().from('Products').select('ProductID,ProductName').take(10),
    //map the appropriate columns to fields property

```

```

    fields: { text: 'ProductName', value: 'ProductID' }
  });
  listObj.appendTo('#listbox');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input id="listbox">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

HTML element

The ListBox can be initialized different tags as described below. Though it is initialized in different tags, the UI appearance and built-in features will behave in the same way.

Select element

When a ListBox is initialized on **SELECT** element, the list items can be assigned through the option tag of the HTML select element.

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
//initiates the component
let listObj: ListBox = new ListBox();

```



```
listObj.appendTo('#listbox');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <h4>Select you favorite car:</h4>
    <select id="listbox">
      <option value="1">Hennessey Venom</option>
      <option value="2">Bugatti Chiron</option>
      <option value="3">Bugatti Veyron Super Sport</option>
      <option value="4">SSC Ultimate Aero</option>
      <option value="5">Koenigsegg CCR</option>
      <option value="6">McLaren F1</option>
      <option value="7">Aston Martin One- 77</option>
      <option value="8">Jaguar XJ220</option>
      <option value="9">McLaren P1</option>
      <option value="10">Ferrari LaFerrari</option>
    </select>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

UL element

The ListBox can be initialized through `` element which contains a collection of `` element. The `` items act as a popup list items of the ListBox. The inner text of the `` element is considered both as text and value fields.

INDEX.TS

```
import { ListBox } from '@syncfusion/ej2-dropdowns';
//initiates the component
let listObj: ListBox = new ListBox();
listObj.appendTo('#listbox');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <h4>Select you favorite car:</h4>
    <ul id="listbox">
      <li>Hennessey Venom</li>
      <li>Bugatti Chiron</li>
      <li>Bugatti Veyron Super Sport</li>
      <li>SSC Ultimate Aero</li>
      <li>Koenigsegg CCR</li>
      <li>McLaren F1</li>
      <li>Aston Martin One- 77</li>
      <li>Jaguar XJ220</li>
      <li>McLaren P1</li>
      <li>Ferrari LaFerrari</li>
    </ul>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to render icons to the list box](#)
- [How to render group based list box](#)
- [How to add items to the list box](#)
- [How to scroll the items in the list box](#)

Icons and templates in EJ2 JavaScript List box control

Icons

To place the icon on a list box, set the [iconCss](#) property to **e-icons** with the required icon CSS. By default, the icon is positioned to the left side of the list.

In the following sample, icon classes are mapped with **iconCss** field.

INDEX.TS

```

import { ListBox, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
ListBox.Inject(CheckBoxSelection);
// define the array of object
let data: { [key: string]: Object }[] = [
  { text: 'Account Settings', iconCss: 'e-list-icons e-list-user-settings' },
  { text: 'Address Book', iconCss: 'e-list-icons e-list-address-book' },
  { text: 'Delete', iconCss: 'e-list-icons e-list-delete' },
  { text: 'Forward', iconCss: 'e-list-icons e-list-forward' },
  { text: 'Reply', iconCss: 'e-list-icons e-list-reply' },
  { text: 'Reply All', iconCss: 'e-list-icons e-list-reply-all' },
  { text: 'Save All Attachments', iconCss: 'e-list-icons e-list-save-all-attachments' },
  { text: 'Save As', iconCss: 'e-list-icons e-list-icon-save-as' },
  { text: 'Touch/Mouse Mode', iconCss: 'e-list-icons e-list-touch' },
  { text: 'Undo', iconCss: 'e-list-icons e-list-undo' }
];
// initialize ListBox component
let listObj: ListBox = new ListBox({
  //set the data to dataSource property
  dataSource: data,
  fields: { text: 'text', iconCss: 'iconCss' }
});
listObj.appendTo('#listbox');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input id="listbox">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Templates

ListBox items can be customized according to the requirement using [itemTemplate](#) property.

In the following sample, the items in the cart are displayed using list box template,

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
// dataSource definition
let data: { [key: string]: Object; }[] = [
    { text: 'JavaScript', pic: 'javascript', description: 'It is a
lightweight interpreted or JIT-compiled programming language.' },
    { text: 'TypeScript', pic: 'typescript', description: 'It is a typed
superset of Javascript that compiles to plain JavaScript.' },
    { text: 'Angular', pic: 'angular', description: 'It is a TypeScript-
based open-source web application framework.' },
    { text: 'React', pic: 'react', description: 'A JavaScript library for
building user interfaces. It can also render on the server using Node.' },
    { text: 'Vue', pic: 'vue', description: 'A progressive framework for
building user interfaces. it is incrementally adoptable.' }
];
// initialize ListBox component
let listBoxObj: ListBox = new ListBox({
    // Set the data source property.

```

```

    dataSource: data,
    // set the template content for list items
    itemTemplate: '<div class="list-wrapper">' +
        '<span class="{pic} e-avatar e-avatar-xlarge e-avatar-circle"></span>'
    +
        '<span class="text">${text}</span><span class="description">' +
        '${description}</span></div>'
    });
listBoxObj.appendTo('#listbox');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 ListBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="listbox">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to enable or disable items in the list box](#)

Selection in EJ2 JavaScript List box control

The ListBox provides support to select an item or a group of item by mouse or keyboard action. There are two selection modes available in list box,

- Single - To select single item in the list box.
- Multiple - To select multiple items in the list box.

On selection of each list box item, [change](#) event is triggered.

Single selection

To enable single selection in the list box, [mode](#) should be set as **Single** in [selectionSettings](#) property.

INDEX.TS

```
import { ListBox } from '@syncfusion/ej2-dropdowns';
// define the array of object
let data: { [key: string]: Object }[] = [
  { text: 'Hennessey Venom' },
  { text: 'Bugatti Chiron' },
  { text: 'Bugatti Veyron Super Sport' },
  { text: 'SSC Ultimate Aero' },
  { text: 'Koenigsegg CCR' },
  { text: 'McLaren F1' },
  { text: 'Aston Martin One- 77' },
  { text: 'Jaguar XJ220' },
  { text: 'McLaren P1' },
  { text: 'Ferrari LaFerrari' }
];
// initialize ListBox component
let listObj: ListBox = new ListBox({
  //set the data to dataSource property
  dataSource: data,
  selectionSettings: {
    mode: 'Single'
  }
});
listObj.appendTo('#listbox');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <h4>Select your favorite car:</h4>
        <input id="listbox">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple selection

To enable multiple selection in the list box, **mode** should be set as **Multiple** in **selectionSettings** property.

To select multiple items, use the SHIFT, CTRL, and arrow keys to make selections.

By default, the selection mode is set as **Multiple**.

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
// define the array of object
let data: { [key: string]: Object }[] = [
    { text: 'Hennessey Venom' },
    { text: 'Bugatti Chiron' },
    { text: 'Bugatti Veyron Super Sport' },
    { text: 'SSC Ultimate Aero' },
    { text: 'Koenigsegg CCR' },
    { text: 'McLaren F1' },
    { text: 'Aston Martin One- 77' },
    { text: 'Jaguar XJ220' },
    { text: 'McLaren P1' },
    { text: 'Ferrari LaFerrari' }
];
// initialize ListBox component
let listObj: ListBox = new ListBox({
    //set the data to dataSource property
    dataSource: data,
    selectionSettings: {
        mode: 'Multiple'
    }
});
listObj.appendTo('#listbox');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <h4>Select your favorite car:</h4>
    <input id="listbox">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Checkbox selection

The ListBox supports checkbox in default and grouped list box which is used to select multiple items. CheckBox selection can be enabled by injecting `CheckBoxSelection` module and also [showCheckBox](#) property should be set as `true`.

Select All

To select all the items in the list box, [showSelectAll](#) should be set as `true`.

INDEX.TS

```

import { ListBox, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
ListBox.Inject(CheckBoxSelection);
// define the array of object
let data: { [key: string]: Object }[] = [
  { text: 'Hennessey Venom' },
  { text: 'Bugatti Chiron' },
  { text: 'Bugatti Veyron Super Sport' },

```



```

    { text: 'SSC Ultimate Aero' },
    { text: 'Koenigsegg CCR' },
    { text: 'McLaren F1' },
    { text: 'Aston Martin One- 77' },
    { text: 'Jaguar XJ220' },
    { text: 'McLaren P1' },
    { text: 'Ferrari LaFerrari' }
  ];
  // initialize ListBox component
  let listObj: ListBox = new ListBox({
    //set the data to dataSource property
    dataSource: data,
    selectionSettings: {
      showCheckbox: true,
      showSelectAll: true
    }
  });
  listObj.appendTo('#listbox');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <h4>Select your favorite car:</h4>
    <input id="listbox">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

To select all the items in the list box, [selectAll](#) method can also be used.

See Also

- [How to select items in list box](#)

Sorting and grouping in EJ2 JavaScript List box control

Sorting

The ListBox supports sorting of available items in the alphabetical order that can be either ascending or descending. This can be achieved using

[sortOrder](#) property. Sort order can be **None**, **Ascending** or **Descending**.

In the following example, the **SortOrder** is set as **Descending**.

INDEX.TS

```
import { ListBox } from '@syncfusion/ej2-dropdowns';
let countryData: { [key: string]: Object }[] = [
  { "Name": "Australia", "Code": "AU" },
  { "Name": "Bermuda", "Code": "BM" },
  { "Name": "Canada", "Code": "CA" },
  { "Name": "Cameroon", "Code": "CM" },
  { "Name": "Denmark", "Code": "DK" },
  { "Name": "France", "Code": "FR" },
  { "Name": "Finland", "Code": "FI" },
  { "Name": "Germany", "Code": "DE" },
  { "Name": "Hong Kong", "Code": "HK" }
];
//initiates the component
let listObj: ListBox = new ListBox({
  dataSource: countryData,
  fields: { text: 'Name' },
  sortOrder: 'Descending'
});
listObj.appendTo('#listbox');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input id="listbox">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouping

The ListBox supports to wrap the nested element into a group based on its category. The category of each list item can be mapped with

[groupBy](#) field in the data table.

In the following example, vegetables are grouped based on its category.

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
let vegetableData: { [key: string]: Object }[] = [
    { "Vegetable": "Cabbage", "Category": "Leafy and Salad", "Id": "item1"
},
    { "Vegetable": "Spinach", "Category": "Leafy and Salad", "Id": "item2"
},
    { "Vegetable": "Wheat grass", "Category": "Leafy and Salad", "Id":
"item3" },
    { "Vegetable": "Yarrow", "Category": "Leafy and Salad", "Id": "item4" },
    { "Vegetable": "Pumpkins", "Category": "Leafy and Salad", "Id": "item5"
},
    { "Vegetable": "Chickpea", "Category": "Beans", "Id": "item6" },
    { "Vegetable": "Green bean", "Category": "Beans", "Id": "item7" },
    { "Vegetable": "Horse gram", "Category": "Beans", "Id": "item8" },
    { "Vegetable": "Garlic", "Category": "Bulb and Stem", "Id": "item9" },
    { "Vegetable": "Nopal", "Category": "Bulb and Stem", "Id": "item10" },
    { "Vegetable": "Onion", "Category": "Bulb and Stem", "Id": "item11" }
];
//initiates the component
let listObj: ListBox = new ListBox({
    dataSource: vegetableData,
    // Map the appropriate columns to fields property along with groupBy
option.

```

```
fields: { groupBy: 'Category', text: 'Vegetable', value: 'Id' }
});
listObj.appendTo('#listbox');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input id="listbox">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

HTML element

The ListBox also supports grouping of list items under specific groups by initiating the `<select>` element using `optgroup`. The nested items are wrapped based on the `<optgroup>` tag that is presents in the `<select>` element.

INDEX.TS

```
import { ListBox } from '@syncfusion/ej2-dropdowns';
//initiates the component
let listObj: ListBox = new ListBox({});
listObj.appendTo('#listbox');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <select id="listbox">
      <optgroup label="Beans">
        <option value="1">Chickpea</option>
        <option value="2">Green bean</option>
        <option value="3">Horse gram</option>
      </optgroup>
      <optgroup label="Leafy and Salad">
        <option value="4">Cabbage</option>
        <option value="5">Spinach</option>
        <option value="6">Wheat grass</option>
      </optgroup>
    </select>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to bind the data source to the list box](#)
- [How to initialize the list box](#)

Drag and drop in EJ2 JavaScript List box control

The ListBox has support to drag an item or a group of selected items and drop it within the same list box or into another list box.

The elements can be customized on drag and drop by using the following events,

Events	Description
dragStart	Triggers when the selected element is being dragged.
drag	Triggers when the selected element is being dragged.
drop	Triggers when the selected element is being dropped.

Single listbox

To drag and drop an item or group of item within the list box can be achieved by setting [allowDragAndDrop](#) property as `true`.

The following sample illustrates how to drag and drop an item within the same list box by enabling [allowDragAndDrop](#) property.

INDEX.TS

```
import { ListBox } from '@syncfusion/ej2-dropdowns';
let groupA: { [key: string]: Object }[] = [
  { "Name": "Australia", "Code": "AU" },
  { "Name": "Bermuda", "Code": "BM" },
  { "Name": "Canada", "Code": "CA" },
  { "Name": "Cameroon", "Code": "CM" },
  { "Name": "Denmark", "Code": "DK" },
  { "Name": "France", "Code": "FR" },
  { "Name": "Finland", "Code": "FI" },
  { "Name": "Germany", "Code": "DE" },
  { "Name": "Hong Kong", "Code": "HK" }
];
let listObj: ListBox = new ListBox({
  // Set the groupa data to the data source.
  dataSource: groupA,
  // Set allowDragAndDrop as `true`.
  allowDragAndDrop: true,
  // Map the appropriate columns to fields property.
  fields: { text: 'Name' }
});
listObj.appendTo('#listbox');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input id="listbox">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple listbox

To drag and drop an item or group of item between two list boxes can be achieved by setting `allowDragAndDrop` property as `true` and `scope` property should be set to both the list boxes.

In the following sample, the `allowDragAndDrop` property is set as `true` and `scope` is set as `combined-list` to enable drop and drop in both list boxes.

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
let groupA: { [key: string]: Object }[] = [
    { "Name": "Australia", "Code": "AU" },
    { "Name": "Bermuda", "Code": "BM" },
    { "Name": "Canada", "Code": "CA" },
    { "Name": "Cameroon", "Code": "CM" },
    { "Name": "Denmark", "Code": "DK" },
    { "Name": "France", "Code": "FR" },
    { "Name": "Finland", "Code": "FI" },
    { "Name": "Germany", "Code": "DE" },
    { "Name": "Hong Kong", "Code": "HK" }
];
let groupB: { [key: string]: Object }[] = [
    { "Name": "India", "Code": "IN" },
    { "Name": "Italy", "Code": "IT" },
    { "Name": "Japan", "Code": "JP" },
    { "Name": "Mexico", "Code": "MX" },
    { "Name": "Norway", "Code": "NO" },

```

```

    { "Name": "Poland", "Code": "PL" },
    { "Name": "Switzerland", "Code": "CH" },
    { "Name": "United Kingdom", "Code": "GB" },
    { "Name": "United States", "Code": "US" }
  ];
  let listObj: ListBox = new ListBox({
    scope: 'combined-list',
    // Set the groupa data to the data source.
    dataSource: groupA,
    // Set allowDragAndDrop as `true`.
    allowDragAndDrop: true,
    height: '290px',
    // Map the appropriate columns to fields property.
    fields: { text: 'Name' }
  });
  listObj.appendTo('#listbox1');
  listObj = new ListBox({
    scope: 'combined-list',
    // Set the groupa data to the data source.
    dataSource: groupB,
    // Set allowDragAndDrop as `true`.
    allowDragAndDrop: true,
    height: '290px',
    // Map the appropriate columns to fields property.
    fields: { text: 'Name' }
  });
  listObj.appendTo('#listbox2');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:45%;">
    <div class="drag-drop-wrapper">

```



```

        <div class="listbox-control1">
            <h4>Group A</h4>
            <input id="listbox1">
        </div>
        <div class="listbox-control2">
            <h4>Group B</h4>
            <input id="listbox2">
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to reorder the items in the list box](#)

Dual list box in EJ2 JavaScript List box control

The dual list box allows the user to move items between two list boxes by clicking the toolbar buttons.

Dual list box can be created by listing items in the [toolbarSettings](#) along with the [scope](#) property.

The following operations can be performed in dual list box,

Options	Description
moveUp	Move the selected item in the upward direction within the list box.
moveDown	Move the selected item in the downward direction within the list box.
moveTo	Move the selected item to the another list box.
moveFrom	Move the selected item from one list box to the another list box.
moveAllTo	Move all the items to the another list box.
moveAllFrom	Move all the items from one list box to the another list box.

The following example illustrates how to move items from Group A to Group B list box.

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
let groupA: { [key: string]: Object }[] = [
    { "Name": "Australia", "Code": "AU" },
    { "Name": "Bermuda", "Code": "BM" },
    { "Name": "Canada", "Code": "CA" },
    { "Name": "Cameroon", "Code": "CM" },
    { "Name": "Denmark", "Code": "DK" },
    { "Name": "France", "Code": "FR" },

```

```

    { "Name": "Finland", "Code": "FI" },
    { "Name": "Germany", "Code": "DE" },
    { "Name": "Hong Kong", "Code": "HK" }
];
let groupB: { [key: string]: Object }[] = [
    { "Name": "India", "Code": "IN" },
    { "Name": "Italy", "Code": "IT" },
    { "Name": "Japan", "Code": "JP" },
    { "Name": "Mexico", "Code": "MX" },
    { "Name": "Norway", "Code": "NO" },
    { "Name": "Poland", "Code": "PL" },
    { "Name": "Switzerland", "Code": "CH" },
    { "Name": "United Kingdom", "Code": "GB" },
    { "Name": "United States", "Code": "US" }
];
let listObj: ListBox = new ListBox({
    // Set the groupA data to the data source.
    dataSource: groupA,
    // Map the appropriate columns to fields property.
    fields: { text: 'Name' },
    // Set the scope of the ListBox.
    scope: '#listbox2',
    // Set the tool settings with set of items.
    toolbarSettings: { items: ['moveUp', 'moveDown', 'moveTo', 'moveFrom',
'moveAllTo', 'moveAllFrom'] },
    height: '330px'
});
listObj.appendTo('#listbox1');
listObj = new ListBox({
    // Set the groupB data to the data source.
    dataSource: groupB,
    // Map the appropriate columns to fields property.
    fields: { text: 'Name' },
    height: '330px'
});
listObj.appendTo('#listbox2');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:45%;">
        <div class="dual-list-wrapper">
            <div class="dual-list-groupa">
                <h4>Group A</h4>
                <input id="listbox1">
            </div>
            <div class="dual-list-groupb">
                <h4>Group B</h4>
                <input id="listbox2">
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in EJ2 JavaScript List box control

The ListBox component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the ListBox component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The ListBox component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the ListBox component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the ListBox component wrapper element as **listbox**, the **UL** element as **presentation**, and its list item as **option**. |

| **aria-label** | Provides an accessible name for the ListBox component. |

| **aria-multiselectable** | Applied to the element with the ListBox role, tells assistive technologies that the list supports multiple selection. The default value is true. |

| **aria-selected** | Applied to elements with role option that are visually styled as selected to inform assistive technologies that the options are selected. |

Keyboard interaction

The ListBox component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the ListBox component.

| Press | To do this |

| --- | --- |

| **Up arrow** | Moves focus to the previous option. |

- | Down arrow | Moves focus to the next option. |
- | Home | Moves focus to first option. |
- | End | Moves focus to last option. |
- | Space | Changes the selection state of the focused option. |
- | Ctrl + A | Selects all options in the list. |
- | Ctrl + Shift + Home | Selects the focused option and all options up to the first option. |
- | Ctrl + Shift + End | Selects the focused option and all options down to the last option. |
- | Ctrl + (Up or Down) | Press Ctrl key with up / down arrow or mouse to select multiple items. |

Ensuring accessibility

The ListBox component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the ListBox component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the ListBox component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

Style and appearance in EJ2 JavaScript List box control

To modify the ListBox appearance, you need to override the default CSS of ListBox component. Please find the list of CSS classes and its corresponding section in ListBox component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class | Purpose of Class

- |.e-listbox-wrapper|To customize the listbox wrapper
- |.e-list-parent .e-list-item|To customize the listbox list items
- |.e-list-parent .e-list-item:hover|To customize the listbox list items on hover
- |.e-list-parent .e-list-item.e-selected|To customize the listbox selected list item
- |.e-listboxtool-wrapper .e-listbox-tool|To customize the listbox toolbar
- |.e-listboxtool-wrapper .e-listbox-tool .e-btn|To customize the listbox toolbar button
- |.e-listboxtool-wrapper .e-listbox-tool .e-btn .e-btn-icon.e-icons::before|To customize the listbox toolbar icon

Horizontal ListBox

You can use [cssClass](#) property to display the Listbox horizontally.

INDEX.TS

```
import { ListBox } from '@syncfusion/ej2-dropdowns';
// define the array of object
let data: { [key: string]: Object }[] = [
  { text: 'Hennessey Venom' },
```

```

    { text: 'Bugatti Chiron' },
    { text: 'Bugatti Veyron Super Sport' },
    { text: 'SSC Ultimate Aero' },
    { text: 'Koenigsegg CCR' },
    { text: 'McLaren F1' },
    { text: 'Aston Martin One- 77' },
    { text: 'Jaguar XJ220' },
    { text: 'McLaren P1' },
    { text: 'Ferrari LaFerrari' },
  ];
  // initialize ListBox component
  let listObj: ListBox = new ListBox({
    //set the data to dataSource property
    dataSource: data,
    cssClass: 'e-horizontal-listbox'
  });
  listObj.appendTo('#listbox');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <h4>Select your favorite car:</h4>
    <input id="listbox">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

How To

Enable scroller in EJ2 JavaScript List box control

The ListBox supports scrolling and it can be achieved by restricting the height of the list box using [height](#) property.

In the following sample, `height` of the list box is restricted to `290px`.

INDEX.TS

```
import { ListBox } from '@syncfusion/ej2-dropdowns';
// define the array of object
let data: { [key: string]: Object }[] = [
  { text: 'Account Settings', id: 'list-01' },
  { text: 'Address Book', id: 'list-02' },
  { text: 'Delete', id: 'list-03' },
  { text: 'Forward', id: 'list-04' },
  { text: 'Reply', id: 'list-05' },
  { text: 'Reply All', id: 'list-06' },
  { text: 'Save All Attachments', id: 'list-07' },
  { text: 'Save As', id: 'list-08' },
  { text: 'Touch/Mouse Mode', id: 'list-09' },
  { text: 'Undo', id: 'list-10' }
];
// initialize ListBox component
let listObj: ListBox = new ListBox({
  //set the data to dataSource property
  dataSource: data,
  height: '290px'
});
listObj.appendTo('#listbox');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
```

```

</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input id="listbox">
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add items in EJ2 JavaScript List box control

To add an item or multiple items, [addItem](#) method can be used. In the following example, the Bugatti Veyron Super Sport and SSC Ultimate Aero items will be added while clicking Add Items button.

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
// define the array of object
let data: { [key: string]: Object }[] = [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' }
];
// initialize ListBox component
let listObj: ListBox = new ListBox({
    //set the data to dataSource property
    dataSource: data
});
listObj.appendTo('#listbox');
document.getElementById('additem').onclick = () => {
    if (!listObj.getDataByValue('Bugatti Veyron Super Sport')) {
        listObj.addItems([
            { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
            { text: 'SSC Ultimate Aero', id: 'list-04' }
        ]);
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 ListBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <button id="additem" style="margin: 10px 0" class="e-btn">Add
Items</button>
        <input id="listbox">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable or disable items in EJ2 JavaScript List box control

To enable or disable items in the list box, [enableItems](#) method can be used. In the following example, the Bugatti Veyron Super Sport and SSC Ultimate Aero items are disabled by default and by clicking Enable Items buttons, the disabled items will be enabled.

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
// define the array of object
let data: { [key: string]: Object }[] = [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' }
];
// initialize ListBox component
let listObj: ListBox = new ListBox({
    //set the data to dataSource property
    dataSource: data,

```

```

        created: () => {
            listObj.enableItems(['Bugatti Veyron Super Sport', 'SSC Ultimate
Aero'], false);
        }
    });
    listObj.appendTo('#listbox');
    document.getElementById('enableitem').onclick = () => {
        listObj.enableItems(['Bugatti Veyron Super Sport', 'SSC Ultimate
Aero']);
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 ListBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <button id="enableitem" style="margin: 10px 0" class="e-btn">Enable
Items</button>
        <input id="listbox">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Form submit in EJ2 JavaScript List box control

In the following code snippet, the value that is in selected state will be sent on form submit.

INDEX.TS

```
import { ListBox } from '@syncfusion/ej2-dropdowns';
import { Button } from '@syncfusion/ej2-buttons';
// define the array of object
let data: { [key: string]: Object }[] = [
    { text: 'Hennessey Venom' },
    { text: 'Bugatti Chiron' },
    { text: 'Bugatti Veyron Super Sport' },
    { text: 'SSC Ultimate Aero' },
    { text: 'Koenigsegg CCR' },
    { text: 'McLaren F1' },
    { text: 'Aston Martin One- 77' },
    { text: 'Jaguar XJ220' },
    { text: 'McLaren P1' },
    { text: 'Ferrari LaFerrari' },
];
// initialize ListBox component
let listObj: ListBox = new ListBox({
    //set the data to dataSource property
    dataSource: data
});
listObj.appendTo('#listbox');
let button: Button = new Button({ isPrimary: true, content: 'Submit' });
button.appendTo('#btnElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 ListBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <form>
      <!--element which is going to render the ListBox-->
      <input id="listbox">
      <button id="btnElement"></button>
    </form>
```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Select items in EJ2 JavaScript List box control

In the following example, **Bugatti Chiron** is selected using [selectItems](#) method.

INDEX.TS

```

import { ListBox } from '@syncfusion/ej2-dropdowns';
// define the array of object
let data: { [key: string]: Object }[] = [
    { text: 'Hennessey Venom' },
    { text: 'Bugatti Chiron' },
    { text: 'Bugatti Veyron Super Sport' },
    { text: 'SSC Ultimate Aero' },
    { text: 'Koenigsegg CCR' },
    { text: 'McLaren F1' },
    { text: 'Aston Martin One- 77' },
    { text: 'Jaguar XJ220' },
    { text: 'McLaren P1' },
    { text: 'Ferrari LaFerrari' }
];
// initialize ListBox component
let listObj: ListBox = new ListBox({
    //set the data to dataSource property
    dataSource: data,
    created: () => {
        listObj.selectItems(['Bugatti Chiron'])
    }
});
listObj.appendTo('#listbox');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 ListBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input id="listbox">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ListView

Getting started in EJ2 JavaScript Listview control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

control Initialization

The Essential JS 2 JavaScript controls can be initialized by using either of the following ways.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

Using local script and style references in a HTML page

Step 1: Create an app folder **myapp** for Essential JS 2 JavaScript controls.

Step 2: You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

Syntax:

Script: **(installed location)\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web (Essential JS 2)\JavaScript\{PACKAGENAME}\dist\global\{PACKAGE_NAME}.min.js**

Styles: **(installed location)\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web (Essential JS 2)\JavaScript\{PACKAGENAME}\styles\material.css**

Example:

Script: C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\16.3.0.17\Web (Essential JS 2)\JavaScript\ej2-lists\dist\global\ej2-lists.min.js

Styles: C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\16.3.0.17\Web (Essential JS 2)\JavaScript\ej2-lists\styles\material.css

The below located script and style file contains all Syncfusion JavaScript (ES5) UI control resources in a single file.

Scripts: **(installed location)**\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASE_VERSION}\Web (Essential JS 2)\JavaScript\ej2\dist\ej2.min.js

Styles: **(installed location)**\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASE_VERSION}\Web (Essential JS 2)\JavaScript\ej2\material.css

Step 3: Create a folder **myapp/resources** and copy/paste the global scripts and styles from the above installed location to **myapp/resources** location.

Step 4: Create a HTML page (index.html) in **myapp** location and add the Essentials JS 2 script and style references.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/lists/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 List's global and dependent scripts -->
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-lists.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Step 5: Now, add the **ListView** element and initiate the **Essential JS 2 ListView** control in the **index.html** by using following code

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/lists/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 List's global and dependent scripts -->
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-lists.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element -->
<div id='element'></div>
<script>
//define the array of string
var arts = ["Artwork", "Abstract", "Modern Painting", "Ceramics", "Animation Art", "Oil Painting"];
// Initialize Essential JS 2 JavaScript ListView control
var listViewInstance = new ej.lists.ListView({
//set the data to datasource property
dataSource: arts
});
//Render initialized ListView
listviewInstance.appendTo("#element");
</script>
</body>
</html>
`
```

Step 6: Now, run the `index.html` in web browser, it will render the **Essential JS 2 ListView** control.

Using [CDN link for script and style reference](#)

Step 1: Create an app folder `myapp` for the Essential JS 2 JavaScript controls.

Step 2: The Essential JS 2 control's global scripts and styles are already hosted in the below CDN link formats.

Syntax:

Script: <https://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js>

Styles: https://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css

Example:

Script: <https://cdn.syncfusion.com/ej2/ej2-lists/dist/global/ej2-lists.min.js>

Styles: <https://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css>

Step 3: Create a HTML page (index.html) in myapp location and add the CDN link references. Now, add the **ListView** element and initiate the **Essential JS 2 List** control in the index.html by using following code.

INDEX.HTML

```
<html><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body><script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Step 4: Now, run the **index.html** in web browser, it will render the **Essential JS 2 ListView** control.

Need to refer dependency control styles and scripts as like above example. We can also use [CRG](#) to generate combined control styles.

See Also

[Data Binding Types](#)

[ListView customization](#)

[Virtualization](#)

Note: You can refer to our [JavaScript ListView](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript ListView Example](#) that show you how to render and configure the ListView in JavaScript.

Data binding in EJ2 JavaScript Listview control

ListView provides an option to load the data either from local dataSource or remote data services. This can be done through the dataSource property that supports the data type of array or [DataManager](#).

ListView supports different kind of data services such as OData, OData V4, and Web API, and data formats like XML, JSON, and, JSONP with the help of DataManager Adaptors.

Fields	Type	Description
id	string	Specifies ID attribute of list item, mapped in dataSource.
text	string	Specifies list item display text field.
isChecked	boolean	Specifies checked status of list item.
isVisible	boolean	Specifies visibility state of list item.
enabled	boolean	Specifies enabled state of list item.
iconCss	string	Specifies the icon class of each list item that will be added before to the list item text.
child	string	Specifies child dataSource fields.
tooltip	string	Specifies tooltip title text field.
groupBy	string	Specifies category of each list item.
sortBy	string	Specifies sorting field, that is used to sort the listview data.
htmlAttributes	string	Specifies list item html attributes field.

When complex data bind to ListView, you should map the fields properly. Otherwise, the ListView properties remain as undefined or null.

Bind to local data

Local data can be represented in two ways, they are as follows:

- Array of simple data.
- Array of JSON data.

Array of simple data

ListView supports to load the array of primitive data like string and numbers. Here, both value and text field act as same.

INDEX.TS

```
import { ListView } from '@syncfusion/ej2-lists';
//define the array of string
let arts: string[] = ["Artwork", "Abstract", "Modern Painting", "Ceramics",
"Animation Art", "Oil Painting"];
//Initialize ListView control
let listViewInstance: ListView = new ListView({
    //set the data to datasource property
    dataSource: arts
});
//Render initialized ListView
```

```
listviewInstance.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Array of JSON data

ListView can generate its list items through an array of complex data. To get it work properly, you should map the appropriate columns to the field property.

In the following example, role column is mapped with the text field.

INDEX.TS

```
import { ListView } from '@syncfusion/ej2-lists';
//define the array of JSON
let settings: { [key: string]: Object }[] = [
  {
    'Name': 'Display',
    'id': 'list-01'
  },
  {
    'Name': 'Notification',
    'id': 'list-02'
  },
];
```

```

    {
        'Name': 'Sound',
        'id': 'list-03'
    },
    {
        'Name': 'Apps',
        'id': 'list-04'
    },
    {
        'Name': 'Storage',
        'id': 'list-05'
    },
    {
        'Name': 'Battery',
        'id': 'list-06'
    }
];
//Initialize ListView control
let listViewInstance: ListView = new ListView({
    //set the data to datasource property
    dataSource: settings,
    //map the appropriate columns to fields property
    fields: { text: 'Name', tooltip: 'Name', id:'id'},
    //set the header tittle for the list
    headerTitle: 'Device settings',
    showHeader: true
});
//Render initialized ListView
listviewInstance.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Bind to remote data

The ListView supports to retrieve the data from remote data services with the help of DataManager control. The Query property allows to fetch data and return it to the ListView from the database.

In the following sample, first 10 employees from the ListView table are displayed.

INDEX.TS

```
import { ListView } from '@syncfusion/ej2-lists';
//import DataManager related classes
import { DataManager, Query } from '@syncfusion/ej2-data';
//Initialize ListView control
let listViewInstance: ListView = new ListView({
    //bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.syncfusion.com/js/production/api/',
        crossDomain: true
    }),
    //Initialize query with the Query instance to get specified set of data
    query: new
    Query().from('ListView').select('EmployeeID,FirstName').take(10),
    //Map the appropriate columns to fields property
    fields: { id: 'EmployeeID', text: 'FirstName' },
    //Set header title
    headerTitle: 'Employees',
    //Set true to show header title
    showHeader: true
});
//Render initialized ListView
listviewInstance.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouping in EJ2 JavaScript Listview control

The ListView supports to wrap the nested element into a group based on the category. The category of each list item can be mapped with `groupBy` field in the data table, that also support single-level navigation.

In the following sample, The cars are grouped based on its category by using the `groupBy` field.

INDEX.TS

```

import { ListView } from '@syncfusion/ej2-lists';
//define the array of JSON
let cars: { [key: string]: Object }[] = [
    {
        'text': 'Audi A4',
        'id': '9bdb',
        'category': 'Audi'
    },
    {
        'text': 'Audi A5',
        'id': '4589',
        'category': 'Audi'
    },
    {
        'text': 'Audi A6',
        'id': 'e807',
        'category': 'Audi'
    },
    {
        'text': 'Audi A7',
        'id': 'a0cc',
        'category': 'Audi'
    },
    {
        'text': 'Audi A8',
        'id': '5e26',
        'category': 'Audi'
    },
],

```

```

    {
        'text': 'BMW 501',
        'id': 'f849',
        'category': 'BMW'
    },
    {
        'text': 'BMW 502',
        'id': '7aff',
        'category': 'BMW'
    },
    {
        'text': 'BMW 503',
        'id': 'b1da',
        'category': 'BMW'
    },
    {
        'text': 'BMW 507',
        'id': 'de2f',
        'category': 'BMW'
    },
    {
        'text': 'BMW 3200',
        'id': 'b2b1',
        'category': 'BMW'
    }
];
//Initialize ListView control
let listViewInstance: ListView = new ListView({
    //set the data to datasource property
    dataSource: cars,
    // map the groupBy field with category column
    fields: { groupBy: 'category', tooltip: 'text' }
});
//Render initialized ListView
listviewInstance.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

The grouping header can be customized by using the groupTemplate property for both inline and fixed group header. The complete customization description and explanation with an example is given in the following link.

[Group Template.](#)

Check list in EJ2 JavaScript Listview control

The ListView supports checkbox in default and group-lists which is used to select multiple items. The checkbox can be enabled by the `showCheckBox` property.

The Checkbox will be useful in the scenario where we need to select multiple options. For Example, In Shipping cart we can be able to select or unselect the desired items before checkout and also it will be useful in selecting multiple items that belongs to the same category using the group list.

INDEX.TS

```

import { ListView } from '@syncfusion/ej2-lists';
let listData: { [key: string]: Object }[] = [
    {text: 'Do Meditation', id: '1'},
    {text: 'Go Jogging', id: '2'},
    {text: 'Buy Groceries', id: '3'},
    {text: 'Pay Phone bill', id: '4'},
    {text: 'Play Football with your friends', id: '5'},
];
// Initialize the ListView control
let listObj: ListView = new ListView({
    //Set the defined data to the data source property
    dataSource: listData,
    headerTitle: 'To DO List',
    showHeader: true,
    //Set true to enable CheckBox
    showCheckBox: true
});
//Render the initialized ListView control
listObj.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Essential JS 2 for ListView </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for ListView UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Checkbox Position

In ListView the checkbox can be positioned into either **Left** or **Right** side of the list-item text. This can be achieved by **checkboxPosition** property. By default, checkbox will be positioned to **Left** of list-item text.

INDEX.TS

```

import { ListView } from '@syncfusion/ej2-lists';
//define the array of JSON
let data: { [key: string]: Object }[] = [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
//Initialize ListView control
let listViewInstance: ListView = new ListView({

```



```

//Set the data to datasource property
dataSource: data,
//Enable checkbox
showCheckBox: true,
//Set Checkbox position
checkBoxPosition: 'Right'
});
//Render initialized ListView
listviewInstance.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Nested list in EJ2 JavaScript Listview control

The ListView control supports Nested list. For that, the child property should be defined for the nested list in the array of JSON.

INDEX.TS

```

import { ListView } from '@syncfusion/ej2-lists';
//define the array of JSON

```

```

let continent: { [key: string]: Object }[] = [
    {
        'text': 'Asia',
        'id': '01',
        'category': 'Continent',
        'child': [{
            'text': 'India',
            'id': '1',
            'category': 'Asia',
            'child': [{
                'text': 'Delhi',
                'id': '1001',
                'category': 'India',
            },
            {
                'text': 'Kashmir',
                'id': '1002',
                'category': 'India',
            },
            {
                'text': 'Goa',
                'id': '1003',
                'category': 'India',
            },
        ]
    },
    {
        'text': 'China',
        'id': '2',
        'category': 'Asia',
        'child': [{
            'text': 'Zhejiang',
            'id': '2001',
            'category': 'China',
        },
        {
            'text': 'Hunan',
            'id': '2002',
            'category': 'China',
        },
        {
            'text': 'Shandong',
            'id': '2003',
            'category': 'China',
        },
    ]
    }
],
    {
        'text': 'North America',
        'id': '02',
        'category': 'Continent',
        'child': [{
            'text': 'USA',
            'id': '3',
            'category': 'North America',
            'child': [{
                'text': 'California',
            },
        ]
    },

```

```

        'id': '3001',
        'category': 'USA',
    },
    {
        'text': 'New York',
        'id': '3002',
        'category': 'USA',
    },
    {
        'text': 'Florida',
        'id': '3003',
        'category': 'USA',
    }
  ]
},
{
  'text': 'Canada',
  'id': '4',
  'category': 'North America',
  'child': [{
    'text': 'Ontario',
    'id': '4001',
    'category': 'Canada',
  },
  {
    'text': 'Alberta',
    'id': '4002',
    'category': 'Canada',
  },
  {
    'text': 'Manitoba',
    'id': '4003',
    'category': 'Canada',
  }
  ]
}]
},
{
  'text': 'Europe',
  'id': '03',
  'category': 'Continent',
  'child': [{
    'text': 'Germany',
    'id': '5',
    'category': 'Europe',
    'child': [{
      'text': 'Berlin',
      'id': '5001',
      'category': 'Germany',
    },
    {
      'text': 'Bavaria',
      'id': '5002',
      'category': 'Germany',
    },
    {
      'text': 'Hesse',
      'id': '5003',
      'category': 'Germany',
    }
  ]
  },
  {
    'text': 'Hesse',
    'id': '5003',
    'category': 'Germany',
  }
  ]
}

```

```

    ]]
  }, {
    'text': 'France',
    'id': '6',
    'category': 'Europe',
    'child': [{
      'text': 'Paris',
      'id': '6001',
      'category': 'France',
    },
    {
      'text': 'Lyon',
      'id': '6002',
      'category': 'France',
    },
    {
      'text': 'Marseille',
      'id': '6003',
      'category': 'France',
    }
  ]
}]
},
{
  'text': 'Australia',
  'id': '04',
  'category': 'Continent',
  'child': [{
    'text': 'Australia',
    'id': '7',
    'category': 'Australia',
    'child': [{
      'text': 'Sydney',
      'id': '7001',
      'category': 'Australia',
    },
    {
      'text': 'Melbourne',
      'id': '7002',
      'category': 'Australia',
    },
    {
      'text': 'Brisbane',
      'id': '7003',
      'category': 'Australia',
    }
  ]
}],
{
  'text': 'New Zealand',
  'id': '8',
  'category': 'Australia',
  'child': [{
    'text': 'Milford Sound',
    'id': '8001',
    'category': 'New Zealand',
  },
  {
    'text': 'Tongariro National Park',
    'id': '8002',
  }
]
}

```

```

        'category': 'New Zealand',
    },
    {
        'text': 'Fiordland National Park',
        'id': '8003',
        'category': 'New Zealand',
    }
  ]
},
{
  'text': 'Africa',
  'id': '05',
  'category': 'Continent',
  'child': [{
    'text': 'Morocco',
    'id': '9',
    'category': 'Africa',
    'child': [{
      'text': 'Rabat',
      'id': '9001',
      'category': 'Morocco',
    },
    {
      'text': 'Toubkal',
      'id': '9002',
      'category': 'Morocco',
    },
    {
      'text': 'Todgha Gorge',
      'id': '9003',
      'category': 'Morocco',
    }
  ]
}, {
  'text': 'South Africa',
  'id': '10',
  'category': 'Africa',
  'child': [{
    'text': 'Cape Town',
    'id': '10001',
    'category': 'South Africa',
  },
  {
    'text': 'Pretoria',
    'id': '10002',
    'category': 'South Africa',
  },
  {
    'text': 'Bloemfontein',
    'id': '10003',
    'category': 'South Africa',
  }
  ]
}
  ]
}
];
//Initialize ListView control
let listViewInstance: ListView = new ListView({
  //set the data to datasource property

```

```

    dataSource: continent,
    // map the groupBy field with category column
    fields: { tooltip: 'text' },
    headerTitle: 'Continent',
    showHeader: true
  });
  //Render initialized ListView
  listViewInstance.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing templates in EJ2 JavaScript Listview control

The ListView control is designed to customize list items, group title and header title. It uses Essential JS2 [Template engine](#) to render the elements.

Header template

ListView header can be customized with the help of the [headerTemplate](#) property.

To customize header template in your application, set your customized template string to [headerTemplate](#) property along with [showHeader](#) property as `true` to display the ListView header.

`ts

```
let listViewInstance: ListView = new ListView({
  data: listData,
  headerTemplate: '<div class="header-content"><span>Header</span></div>',
  showHeader: true
})
,
```

In the following example, we have rendered Listview with customized header which contains search, add and sort buttons.

INDEX.TS

```
import { ListView } from '@syncfusion/ej2-lists';
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let fruitsdata: { [key: string]: Object }[] = [
  { text: 'Date', id: '1', imgUrl: './dates.jpg' },
  { text: 'Fig', id: '2', imgUrl: './fig.jpg' },
  { text: 'Apple', id: '3', imgUrl: './apple.png' },
  { text: 'Apricot', id: '4', imgUrl: './apricot.jpg' },
  { text: 'Grape', id: '5', imgUrl: './grape.jpg' },
  { text: 'Strawberry', id: '6', imgUrl: './strawberry.jpg' },
  { text: 'Pineapple', id: '7', imgUrl: './pineapple.jpg' },
  { text: 'Melon', id: '8', imgUrl: './melon.jpg' },
  { text: 'Lemon', id: '9', imgUrl: './lemon.jpg' },
  { text: 'Cherry', id: '10', imgUrl: './cherry.jpg' },
];
let listViewInstance: any = new ListView({
  dataSource: fruitsdata,
  headerTemplate: '<div class="headerContainer"><span
class="fruitHeader">Fruits</span><button id="search"></button><button
id="add"></button><button id="sort"></button></div>',
  showHeader: true,
  actionComplete: renderHeaderButtons
});
listViewInstance.appendTo('#element');
function renderHeaderButtons() {
  ['search', 'sort', 'add'].forEach((item: string) => {
    new Button({ iconCss: `e-icons e-${item}-icon`, cssClass: 'e-small
e-round', isPrimary: true }, `#${item}`)
  });
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Template

ListView items can be customized with the help of the [template](#) property.

To customize list items in your application, set your customized template string to `template` property.

`ts

```

let listViewInstance: ListView = new ListView({
data: listData,
template: '<div class="list-tem"><span>${text}</span></div>',
})
`

```

We provided the following built-in CSS classes to customize the list-items. Refer to the following table.

CSS class	Description
e-list-template, e-list-wrapper	These classes are used to differentiate normal and template rendering, which are mandatory for template rendering. The <code>e-list-template</code> class should be added to the root of the ListView element and <code>e-list-wrapper</code> class should be added to the template element wrapper

```

new ListView({
  dataSource: data,
  cssClass: 'e-list-template',
  template: '<div class="e-list-wrapper"></div>', '#list');

```


| e-list-content | This class is used to align list content and it should be added to the content element
`

 <div class="e-list-wrapper">
ListItem
</div>|`

| e-list-avatar | This class is used for avatar customization. It should be added to the template element wrapper. After adding it, we can customize our element with [Avatar](#) classes
`

 <div class="e-list-wrappere-list-avatar">
 MR
ListItem
</div>|`

| e-list-avatar-right | This class is used to align avatar to right side of the list item. It should be added to the template element wrapper. After adding it, we can customize our element with [Avatar](#) classes
`

 <div class="e-list-wrappere-list-avatar-right">
 ListItem
MR
</div>|`

| e-list-badge | This class is used for badge customization .It should be added to the template element wrapper. After adding it, we can customize our element with [Badge](#) classes
`

 <div class="e-list-wrappere-list-badge">
 ListItem
MR
</div>|`

| e-list-multi-line | This class is used for multi-line customization. It should be added to the template element wrapper. After adding it, we can customize List item's header and description
`

<div class="e-list-wrappere-list-multi-line">
 ListItem
</div>|`

| e-list-item-header | This class is used to align a list header and it should be added to the header element along with the multi-line class
`

 <div class="e-list-wrappere-list-multi-line">
 ListItem Header
 ListItem
</div>|`

In the following example, we have customized list items with built-in CSS classes.

INDEX.TS

```
import { ListView } from "@syncfusion/ej2-lists";
let template: string = '<div class="e-list-wrapper e-list-multi-line e-list-avatar">' +
    '${if (avatar!=="")}' +
    '<span class="e-avatar e-avatar-circle">${avatar}</span>' +
    '${else}' +
    '<span class="${pic} e-avatar e-avatar-circle"> </span>' +
    '${/if}' +
    '<span class="e-list-item-header">${text}</span>' +
    '<span class="e-list-content">${contact}</span>' +
    '</div>';
//Define an array of JSON data
let dataSource: any = [
    {
        text: "Jenifer",
        contact: "(206) 555-985774",
        id: "1",
        avatar: "",
    }
];
```

```

        pic: "pic01"
    },
    { text: "Amenda", contact: "(206) 555-3412", id: "2", avatar: "A", pic: ""
    },
    {
        text: "Isabella",
        contact: "(206) 555-8122",
        id: "4",
        avatar: "",
        pic: "pic02"
    },
    {
        text: "William ",
        contact: "(206) 555-9482",
        id: "5",
        avatar: "W",
        pic: ""
    },
    {
        text: "Jacob",
        contact: "(71) 555-4848",
        id: "6",
        avatar: "",
        pic: "pic04"
    },
    { text: "Matthew", contact: "(71) 555-7773", id: "7", avatar: "M", pic: ""
    },
    {
        text: "Oliver",
        contact: "(71) 555-5598",
        id: "8",
        avatar: "",
        pic: "pic03"
    },
    {
        text: "Charlotte",
        contact: "(206) 555-1189",
        id: "9",
        avatar: "C",
        pic: ""
    }
    ];
// Initialize the ListView control
let listObj: ListView = new ListView({
    //Set the defined data to the dataSource property
    dataSource: dataSource,
    //Map the appropriate columns to the fields property
    fields: { text: "text" },
    //Set the width of the ListView
    width: "350px",
    //Enable the header of the ListView
    showHeader: true,
    //Set the header title
    headerTitle: "Contacts",
    //set cssClass for template customization
    cssClass: 'e-list-template',
    //Set the customized template

```

```

    template: template,
    sortOrder: "Ascending"
  });
  //Render the initialized ListView control
  listObj.appendTo("#List");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="List" tabindex="1"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Group template

ListView group header can be customized with the help of the [groupTemplate](#) property.

To customize the group template in your application, set your customized template string to [groupTemplate](#) property.

```
`ts
```

```

let listViewInstance: ListView = new ListView({
  data: listData,
  groupTemplate: '<div class="group-header"><span>${items[0].category}</span></div>',

```

```

})
`

```

In the following example, we have grouped Listview based on the category. The category of each list item should be mapped with [groupBy](#) field of the data. We have also displayed grouped list items count in the group list header.

INDEX.TS

```

import { ListView } from '@syncfusion/ej2-lists';
let template: string = '<div class="e-list-wrapper e-list-multi-line e-list-avatar">' +
    '<img class="e-avatar e-avatar-circle" src=${image}
style="background:#BCBCBC" />' +
    '<span class="e-list-item-header">${Name}</span>' +
    '<span class="e-list-content">${contact}</span>' +
    '</div>';
//Define an array of JSON data
let dataSource = [
    { Name: 'Nancy', contact: '(206) 555-985774', id: '1', image:
'https://ej2.syncfusion.com/demos/src/grid/images/1.png', category:
'Experience' },
    { Name: 'Janet', contact: '(206) 555-3412', id: '2', image:
'https://ej2.syncfusion.com/demos/src/grid/images/3.png', category:
'Fresher' },
    { Name: 'Margaret', contact: '(206) 555-8122', id: '4', image:
'https://ej2.syncfusion.com/demos/src/grid/images/4.png', category:
'Experience' },
    { Name: 'Andrew ', contact: '(206) 555-9482', id: '5', image:
'https://ej2.syncfusion.com/demos/src/grid/images/2.png' category:
'Experience' },
    { Name: 'Steven', contact: '(71) 555-4848', id: '6', image:
'https://ej2.syncfusion.com/demos/src/grid/images/5.png', category:
'Fresher' },
    { Name: 'Michael', contact: '(71) 555-7773', id: '7', image:
'https://ej2.syncfusion.com/demos/src/grid/images/6.png', category:
'Experience' },
    { Name: 'Robert', contact: '(71) 555-5598', id: '8', image:
'https://ej2.syncfusion.com/demos/src/grid/images/7.png', category:
'Fresher' },
    { Name: 'Laura', contact: '(206) 555-1189', id: '9', image:
'https://ej2.syncfusion.com/demos/src/grid/images/8.png', category:
'Experience' },
];
// Initialize the ListView control
let listObj: ListView = new ListView({
    //Set the defined data to the data source property
    dataSource: dataSource,
    width: 350,
    //Map the appropriate columns to the fields property
    fields: { text: 'Name', groupBy: 'category' },
    //set cssClass for template customization
    cssClass: 'e-list-template',
    //Set the customized template
    template: template,

```

```

        groupTemplate: '<div><span
class="category">${items[0].category}</span> <span class="count">
${items.length} Item(s)</span></div> '
    });
    //Render the initialized ListView control
    listObj.appendTo('#List');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="col-lg-12 control-section">
            <!-- ListView element -->
            <div id="List" tabindex="1">
                </div>
            </div>
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

We can also render other EJ2 controls in ListView templates by rendering that controls on [actionComplete](#) event. Refer to the below code snippet.

```

`ts
let listViewInstance: ListView = new ListView({
data: listData,

```

```

template: '<div><span>${text}</span><button id="delete"></button></div>',
actionComplete: () => {
  new Button({iconCss: 'e-icons e-delete-icon' cssClass: 'e-small e-round', isPrimary: true }, '#delete')
}
})
,

```

Virtualization in EJ2 JavaScript Listview control

UI virtualization loads only viewable list items in a view port, which will improve the ListView performance while loading a large number of data.

Module injection

To use UI virtualization, you should import `virtualization` module from the `ej2-lists` package and inject it using `ListView.Inject()`

method as shown in the following code snippet.

```

`ts
import { ListView, Virtualization } from '@syncfusion/ej2-lists';
ListView.Inject(Virtualization);
,

```

Getting started

UI virtualization can be enabled in the ListView by setting the [enableVirtualization](#) property to true.

It has two types of scrollers as follows:

Window scroll: This scroller is used in the ListView by default.

Container scroll: This scroller is used, when the height property of the ListView is set.

INDEX.TS

```

import { ListView, Virtualization } from '@syncfusion/ej2-lists';
ListView.Inject(Virtualization);
let listData: { [key: string]: string | object }[] = [];
listData = [
  { text: 'Nancy', id: '0' },
  { text: 'Andrew', id: '1' },
  { text: 'Janet', id: '2' },
  { text: 'Margaret', id: '3' },
  { text: 'Steven', id: '4' },
  { text: 'Laura', id: '5' },
  { text: 'Robert', id: '6' },
  { text: 'Michael', id: '7' },
  { text: 'Albert', id: '8' },
  { text: 'Nolan', id: '9' }
];
for (let i: number = 10; i <= 1010; i++) {
  let index: number = parseInt((Math.random() * 10).toString());
  listData.push({ text: listData[index].text, id: i.toString() });
}

```

```
let listObj: ListView = new ListView({
  //Set defined data to dataSource property.
  dataSource: listData,
  //Set height
  height: 500,
  //enable UI virtualization
  enableVirtualization: true,
});
listObj.appendTo('#ui-list');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="ui-list"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

You can use the **template** property to customize list items in UI virtualization.

INDEX.TS

```
import { ListView, Virtualization } from '@syncfusion/ej2-lists';
ListView.Inject(Virtualization);
let listData: { [key: string]: string | object }[] = [];
listData = [
  { name: 'Nancy', icon: 'N', id: '0', },
  { name: 'Andrew', icon: 'A', id: '1' },
  { name: 'Janet', icon: 'J', id: '2' },
```

```

    { name: 'Margaret', imgUrl:
    '//ej2.syncfusion.com/demos/src/listview/images/margaret.png', id: '3' },
    { name: 'Steven', icon: 'S', id: '4' },
    { name: 'Laura', imgUrl:
    '//ej2.syncfusion.com/demos/src/listview/images/laura.png', id: '5' },
    { name: 'Robert', icon: 'R', id: '6' },
    { name: 'Michael', icon: 'M', id: '7' },
    { name: 'Albert', imgUrl:
    '//ej2.syncfusion.com/demos/src/listview/images/albert.png', id: '8' },
    { name: 'Nolan', icon: 'N', id: '9' }
  ]
  for (let i: number = 10; i <= 1010; i++) {
    let index: number = parseInt((Math.random() * 10).toString());
    listData.push({ name: listData[index].name, icon: listData[index].icon,
    imgUrl: listData[index].imgUrl, id: i.toString() });
  }
  var template: Function = (data: any) => {
    var result = `<div class="e-list-wrapper e-list-avatar" >`+
    `<span class="e-avatar e-avatar-circle ${data.icon} ${data.imgUrl ?
    'hideUI' : 'showUI' }">${data.icon}</span> ` +
    `<span class="e-list-content">${data.name}</span></div>`;
    return result;
  };
  let listObj: ListView = new ListView({
    //Set defined data to dataSource property.
    dataSource: listData,
    //enable UI virtualization
    enableVirtualization: true,
    //Set height
    height: 500,
    //Set header title
    headerTitle: 'Contacts',
    //Set true to show header title
    showHeader: true,
    cssClass: 'e-list-template',

    //Set defined customized template
    template: template
  });
  listObj.appendTo('#ui-list');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
  Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">

```



```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ui-list"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Conditional rendering

The following conditional rendering support is provided for the template and groupTemplate.

| Name | Syntax |

| --- | --- | --- |

| conditional class | `<div class="$ { $id % 2 === 0 ? 'even-list' : 'odd-list' }"></div>` |

| conditional attribute | `<div id="$ { $id % 2 === 0 ? 'even-list' : 'odd-list' }"></div>` |

| conditional text content | `<div>${ $id % 2 === 0 ? 'even-list' : 'odd-list' }</div>` |

In the following sample, the light blue is applied for the even list and light coral is applied for the odd list based on the conditional class.

INDEX.TS

```

import { ListView, Virtualization } from '@syncfusion/ej2-lists';
ListView.Inject(Virtualization);
let listData: { [key: string]: string | object }[] = [];
listData = [
    { text: 'Nancy', id: '0' },
    { text: 'Andrew', id: '1' },
    { text: 'Janet', id: '2' },
    { text: 'Margaret', id: '3' },
    { text: 'Steven', id: '4' },
    { text: 'Laura', id: '5' },
    { text: 'Robert', id: '6' },
    { text: 'Michael', id: '7' },
    { text: 'Albert', id: '8' },
    { text: 'Nolan', id: '9' }
];
for (let i: number = 10; i <= 1010; i++) {
    let index: number = parseInt((Math.random() * 10).toString());

```

```

        listData.push({ text: listData[index].text, id: i.toString() });
    }
    let listObj: ListView = new ListView({
        //Set defined data to dataSource property.
        dataSource: listData,
        //enable UI virtualization
        enableVirtualization: true,
        //Set height
        height: 500,
        //Set defined customized template
        template: '<div id="list-container" class="{ $id % 2 === 0 ? \'even-list\' : \'odd-list\' }" > ${text} </div>'
    });
    listObj.appendTo('#ui-list');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ui-list"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Scrolling in EJ2 JavaScript Listview

Scrolling is a technique that allows you to load more items as the user scrolls through a list, providing a seamless and dynamic user experience.

Render the ListView with [dataSource](#), and bind the [scroll](#) event. Within the scroll event, you can access information such as the scroll direction, event name and the distance from the scrollbar to the top and bottom ends through the distanceY parameter.

In the given example, new data is seamlessly added while scrolling. When you scrolls to the bottom and the distance scrolled is less than 100 pixels, it dynamically loads a batch of items into the list as long as there are more items to render.

INDEX.TS

```
import { ListView, ScrolledEventArgs } from '@syncfusion/ej2-lists';
let data: { [key: string]: Object }[] = [
  {
    text: "Hi Guys, Good morning! \uD83D\uDE0A, I'm very delighted to share with you the news that our team is going to launch a new mobile application",
    positionClass: 'right',
  },
  {
    text: "Oh! That's great \uD83D\uDE42",
    positionClass: 'left',
  },
  {
    text: 'That is a good news \uD83D\uDE00',
    positionClass: 'right',
  },
  {
    text: 'What kind of application we are gonna launch? \uD83E\uDD14',
    positionClass: 'left',
  },
  {
    text: 'A kind of "Emergency Broadcast App" like being able to just invite someone to teams without it impacting how many people have official access',
    positionClass: 'right',
  },
  {
    text: 'Who will be the client users for our applications? ',
    positionClass: 'left',
  },
  {
    text: 'Is currently the only way to invite someone through 0365? Just wondering down the road how organization would want to handle that with freelancers, like being able to just invite someone to teams without it impacting how many people have official access \uD83D\uDE14',
    positionClass: 'right',
  },
  {
    text: 'Yes, however, that feature of inviting someone from outside your organization is planned - expected closer to GA next year \uD83D\uDC4D',
    positionClass: 'left',
  },
  {
    text: 'I guess we should switch things over to hear for a while. How long does the trial last? \uD83E\uDD14',
    positionClass: 'right',
  },
],
```

```

    { text: 'I think the trial is 30 days. \uD83D\uDE03', positionClass:
    'left' },
    {
        text: 'Only 0365 only members of your organization. They said that they
        are listening to customer feedback and hinted that guest users would be
        brought in down the road \uD83D\uDE09',
        positionClass: 'right',
    },
    { text: 'Cool thanks! \uD83D\uDC4C', positionClass: 'left' },
];
function loadTemplate(data: { positionClass: string; text: string; }) {
    var containerClass =
        data.positionClass === 'right' ? 'justify-content: flex-end' : '';
    return (
        '<div style="display: flex; ' +
        containerClass +
        ';" class="e-list-wrapper e-list-multi-line">' +
        '<span style="display: block; white-space: normal;max-width: 80%;
padding: 10px;background-color: #e0e0e0;border-radius: 10px;word-wrap:
break-word;" class="e-list-item-header">' +
        data.text +
        '</span>' +
        '</div>'
    );
}
//Initialize ListView component
let listObj_1: ListView = new ListView({
    //Initialize dataSource with the DataManager instance.
    dataSource: data.slice(0, 5),
    scroll: onListScrolled,
    height: 320,
    width: 400,
    template: loadTemplate,
    cssClass: 'e-list-template',
});
//Render initialized ListView component
listObj_1.appendTo('#list-scrolling-down');
var itemsRendered = 5;
var itemsPerScroll = 5;
var result: { text: string; positionClass: string; }[] ;
function onListScrolled(args: ScrolledEventArgs) {
    if (args.scrollDirection === 'Bottom') {
        if (args.distanceY < 100) {
            if (itemsRendered < data.length) {
                var startIndex = itemsRendered;
                var endIndex = Math.min(itemsRendered + itemsPerScroll,
data.length);
                result = data.slice(startIndex, endIndex) as { text: string;
positionClass: string; }[];
                listObj_1.addItem(result);
                itemsRendered = endIndex;
            }
        }
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="grid-container">
      <div>
        <h3>Chat</h3>
        <div id="list-scrolling-down">
        </div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in EJ2 JavaScript Listview component

The ListView component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the ListView component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | ` |`

| Screen Reader Support | ` |`

| Right-To-Left Support | ` |`

| Color Contrast | ` |`

| Mobile Device Support | ` |`

| Keyboard Navigation Support | ` |`

| [Accessibility Checker](#) Validation | ` |`

| [Axe-core](#) Accessibility Validation | ` |`

`<style>`

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

`</style>`

`<div> - All features of the component meet the requirement.</div>`

`<div> - Some features of the component do not meet the requirement.</div>`

`<div> - The component does not meet the requirement.</div>`

WAI-ARIA attributes

The ListView component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the ListView component:

| Attributes | Purpose |

| --- | --- |

| `role=list` | Specifies the non selectable list container. |

| `role=listitem` | Specifies the role of each item in a selectable ListView and its containment within the list. |

| `role=presentation` | Specifies the role of non selectable list element. |

- | `role=checkbox` | Indicates checkbox control along with listitem element. |
- | `aria-checked` | Indicates the current checked state of checkbox. |
- | `aria-label` | Provides an accessible name for the ListView Checkbox. |
- | `aria-disabled` | Indicates element is perceivable but disabled. |

Keyboard interaction

The ListView component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the ListView component.

| Keyboard shortcuts | Actions |

|-----|-----|

| Arrow Up | Move to the previous list item |

| Arrow Down | Move to the next list item |

| Space | Check and uncheck the targeted list from the whole list |

| BackSpace | Get back to the previous lists if it is nested list |

| Home | Moves focus to first list item |

| End | Moves focus to last list item |

Ensuring accessibility

The ListView component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the ListView component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the ListView component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

Style in EJ2 JavaScript Listview control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

Customizing ListView

Use the following CSS to customize the ListView.

```
`css
.e-listview {
border: 5px solid rgb(173, 255, 47);
}
```

Customizing the list items

Use the following CSS to customize the items of ListView.

```
`css
.e-listview .e-list-item {
text-align: center;
color: pink;
background-color: #2fa1ff;
}
`
```

Customizing ListView's header

Use the following CSS to customize the header of ListView control.

```
`css
.e-listview .e-list-header{
color: #2fa1ff;
justify-content: center;
}
`
```

Customizing group header of ListView

Use the following CSS to customize the category of the group items.

```
`css
.e-listview .e-list-group-item {
color: rgb(173, 255, 47);
background-color: maroon;
text-align: end;
}
`
```

Customizing the hover state of ListView control

Use the following CSS to customize the list item when hovering.

Customizing ListView hover state with the checkbox checked

```
`css
.e-listview .e-list-item.e-hover.e-active.e-checklist {
color: rgb(83, 5, 79);
background-color: rgb(173, 255, 47);
}
`
```


`

Customizing ListView hover state

`css

```
.e-listview .e-list-item.e-hover {
color:red;
background-color: rgb(173, 255, 47);
}
```

`

Customizing selected item of ListView control

Use the following CSS to customize the selected list item.

Customizing ListView's selected item with the checkbox checked

`css

```
.e-listview .e-list-item.e-checklist.e-focused.e-active {
color: rgb(83, 5, 79);
background-color: rgb(0, 15, 100);
}
```

`

Customizing ListView's selected item

`css

```
.e-listview .e-list-item.e-focused {
color: #2fa1ff;
background-color: rgb(0, 15, 100);
}
```

`

Ej1 api migration in EJ2 JavaScript Listview control

This article describes the API migration process of ListView component from Essential JS 1 to Essential JS 2.

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Virtualization | **Property:** *allowVirtualScrolling*
 \$("#list").ejListView ({
 dataSource: dataSrc,
 allowVirtualScrolling: true,
virtualScrollMode: "normal"
 }); | **Property:** *enableVirtualization*
 let list: ListView = new ej.lists.ListView({
 dataSource: data,
 enableVirtualization: true
 });
 list.appendTo('#test'); |

| Checkbox | **Property:** *enableCheckMark*
 \$("#list").ejListView ({
dataSource: dataSrc,
enableCheckMark: true
 }); | **Property:** *showCheckBox*
 let list: ListView = new ej.lists.ListView({
 dataSource: data,
 showCheckBox: true
 });
 list.appendTo('#test'); |

| Fields | **Property:** *fieldSettings*
 enableGroupList [
 <ul data-ej-grouplisttitle="Settings">
]
 \$(" #list").ejListView({
 dataSource: dataSrc,
 fieldSettings:{'text':'textfield'},
 enableGroupList: true
 }); | **Property:** *fields*
 Inner properties: *child, enabled, groupBy htmlAttributes, iconsCss, id, isChecked, isVisible, sortBy, tableName, text, tooltip*
 let list: ListView = new ej.lists.ListView({
 dataSource: data,
 fields: { enabled: enable_item, groupBy: groupByProp,...}
 });
 list.appendTo('#test'); |

| Template | **Property:** *renderTemplate*
 <div id="lb">

 <li data-ej-templateid="target1">
 <li data-ej-templateid="target2">
 <li data-ej-templateid="target3">

 </div>
 <div id="target1">
 <div> Template1 </div>
 </div>
 <div id="target2">
 <div> Template2 </div>
 </div>
 <div id="target3">
 <div> Template3 </div>
 </div>
 \$(" #list").ejListView({
 dataSource: dataSrc,
 renderTemplate: true
 }); | **Property:** *template*
 let list: ListView = new ej.lists.ListView({
 dataSource: data,
 template: "<div>string template</div>"
 });
 list.appendTo('#test'); |

| Animation | **Not Applicable** | **Property:** *animation*
 let list: ListView = new ej.lists.ListView({
 dataSource: data,
 animation: { effect: 'SlideLeft', duration: 400, easing: 'ease' }
 });
 list.appendTo('#test'); |

| Enable | **Not Applicable** | **Property:** *enable*
 let list: ListView = new ej.lists.ListView({
 dataSource: data,
 enable: true
 });
 list.appendTo('#test'); |

| Template for grouping | **Not Applicable** | **Property:** *groupTemplate*
 let list: ListView = new ej.lists.ListView({
 dataSource: data,
 groupTemplate: 'group template string'
 });
 list.appendTo('#test'); |

| Template for header | **Not Applicable** | **Property:** *headerTemplate*
 let list: ListView = new ej.lists.ListView({
 dataSource: data,
 headerTemplate: 'header template string'
 });
 list.appendTo('#test'); |

| HTML attributes | **Not Applicable** | **Property:** *htmlAttributes*
 var attribute = {id: 'listid', class: '.listtest'}
 let list: ListView = new ej.lists.ListView({
 dataSource: data,
 htmlAttributes: this.attribute
 });
 list.appendTo('#test'); |

| Clear | **Method:** *clear()*
 \$(" #list").ejListView({dataSource: dataSrc });
 \$(" #lb").ejListView("clear"); | **Property** *dataSource*
 let emptyDataSrc: { [key: string]: Object }[] = [];
 let list: ListView = new ej.lists.ListView({
 dataSource: emptyDataSrc,
 });
 list.appendTo('#test');
 list.destory();
 |

| IsChecked | **Method:** *isChecked()*
 \$(" #list").ejListView({enableCheckMark:true});
 \$(" #lb").ejListView("isChecked"); | **Method:** *checkItem()*
 let list: ListView = new ej.lists.ListView({
 dataSource: data,
 });
 list.appendTo('#test');
 list.checkItem({id:'2'});
 |

| Load Ajax Content | **Method:** *loadAjaxContent()*
 \$(" #list").ejListView ({ enableAjax: true });
 \$(" #list").ejListView("loadAjaxContent","load1.html"); | **Event:** *onSuccess*
 let template: string;
 let ajax: Ajax = new Ajax('./template.html', 'GET', false);
 ajax.onSuccess= (e: string)=>{
 template = e;
 }
 ajax.send();
 let list: ListView = new ej.lists.ListView({
 dataSource: data,
 template: template,
 headerTitle: 'Settings',
 showHeader: true
 });
 list.appendTo('#element');
 |

```
| Remove CheckMark | Method: removeCheckMark() <br />
$("#list").ejListView({enableCheckMark:true}); <br /> $("#list").ejListView("removeCheckMark",2); |
Method: uncheckItem() <br />let list: ListView = new ej.lists.ListView({<br />dataSource: data,<br
/>});<br />list.appendTo('#test');<br />list.uncheckItem ({id:'2'})<br />|

| Set Active | Method: setActive() <br /> $("#list").ejListView({persistSelection:true}); <br />
$("#list").ejListView("setActive",2); | Method: selectItem() <br /> let list: ListView = new
ej.lists.ListView({<br />dataSource: data,<br />});<br />list.appendTo('#test');<br />list.selectItem({id:'2'})
<br />|

| Select | Not Applicable | Event: select <br /> let list: ListView = new ej.lists.ListView({ <br /> select:
function(e: Event): void { } <br /> }); <br /> list.appendTo('#test');|

| Ajax Before Load | Event: ajaxbeforeLoad <br /> $("#list").ejListView({ <br /> enableAjax: true, <br />
ajaxBeforeLoad: function (args) { } <br /> }); | Event: onSuccess <br /> let template: string;<br /> let ajax:
Ajax = new Ajax('./template.html', 'GET', false);<br /> ajax.onSuccess= (e: string)=>{ <br /> template =
e;<br /> }<br /> ajax.beforeSend();<br /> let list: ListView = new ej.lists.ListView({<br /> dataSource:
data,<br />template: template,<br />headerTitle: 'Settings',<br />showHeader: true<br />});<br
/>list.appendTo('#element');<br />|

| Ajax Complete | Event: ajaxComplete <br /> $("#list").ejListView({ <br /> enableAjax: true, <br />
ajaxComplete: function (args) { } <br /> }); | Event: onSuccess <br /> let template: string;<br /> let ajax:
Ajax = new Ajax('./template.html', 'GET', false);<br /> ajax.onSuccess= (e: string)=>{ <br /> template =
e;<br /> }<br /> ajax.send();<br /> let list: ListView = new ej.lists.ListView({<br /> dataSource: data,<br
/>template: template,<br />headerTitle: 'Settings',<br />showHeader: true<br />});<br
/>list.appendTo('#element');<br />|

| Ajax Error | Event: ajaxError <br /> $("#list").ejListView({ <br />enableAjax: true, <br />ajaxError:
function (args) { } <br />}); | Event: onError <br /> let template: string; <br />let ajax: Ajax = new
Ajax('./template.html', 'GET', false);<br />ajax.onError = (e: string)=>{<br />template = e;<br />}<br
/>ajax.send();<br />let list: ListView = new ej.lists.ListView({<br />dataSource: data,<br />template:
template,<br />headerTitle: 'Settings',<br />showHeader: true<br />});<br
/>list.appendTo('#element');<br />|
```

How To

Get selected items from listview in EJ2 JavaScript Listview control

Single or many items can be selected by users in the ListView control. An API is used to get selected items from the list items. This is called as the [getSelectedItems](#) method.

getSelectedItems method

This is used to get the details of the currently selected item from the list items. It returns the [SelectedItem](#) | [SelectedCollection](#)

The `getSelectedItems` method returns the following items from the selected list items.

Return type	Purpose
text	Returns the text of selected item lists
data	Returns the whole data of selected list items, i.e., returns the fields data of selected li.
item	Returns the collections of list items

INDEX.TS

```

import { ListView, SelectedCollection } from '@syncfusion/ej2-lists';
import { Button } from '@syncfusion/ej2-buttons';
//define the array of string
let data: { [key: string]: Object }[] = [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02', isChecked: true },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04', isChecked: true },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07', isChecked: true },
    { text: 'Jaguar XJ220', id: 'list-08' }
];
//Initialize the ListView control
let listViewInstance: ListView = new ListView({
    //set the data to the dataSource property
    dataSource: data,
    //Enable check box
    showCheckBox: true,
});
//Render the initialized ListView
listviewInstance.appendTo("#element");
let button: Button = new Button();
button.appendTo("#btn")
document.getElementById('btn').addEventListener('click', () => {
    let selecteditem: SelectedCollection =
listviewInstance.getSelectedItems() as SelectedCollection;
    let data: HTMLElement = document.getElementById('val');
    data.innerHTML = "";
    let row1: HTMLTableRowElement = document.createElement('tr');
    let header1: HTMLTableHeaderCellElement = document.createElement('th');
    header1.innerHTML = 'Text';
    row1.appendChild(header1);
    let header2 = document.createElement('th');
    header2.innerHTML = 'Id';
    row1.appendChild(header2);
    document.getElementById('val').appendChild(row1);
    for (let i: number = 0; i < (selecteditem["data"] as { [key: string]:
object }[]).length; i++) {
        let row2: HTMLTableRowElement = document.createElement('tr');
        row2.setAttribute("id", i.toString());
        let data1: HTMLElement = document.createElement('td');
        data1.innerHTML = selecteditem["text"][i];
        row2.appendChild(data1);
        let data2: HTMLElement = document.createElement('td');
        data2.innerHTML = (selecteditem["data"] as { [key: string]: object
}[])[i].id.toString();
        row2.appendChild(data2);
        document.getElementById('val').appendChild(row2);
    }
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <div id="text">
      <button id="btn">Get selected Items</button>
      <table id="val">
        </table>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Add and remove list items from listview in EJ2 JavaScript Listview control

You can add or remove list items from the ListView control using the [addItem](#) and [removeItem](#) methods. Refer to the following steps to add or remove a list item.

- Render the ListView with data source, and use the [template](#) property to append the delete icon for each list item. Also, bind the click event for the delete icon using the [actionComplete](#) handler.
- Render the Add Item button, and bind the click event. On the click event handler, pass data with random id to the [addItem](#) method to add a new list item on clicking the Add Item button.
- Bind the click handler to the delete icon created in step 1. Within the click event, remove the list item by passing the delete icon list item to [removeItem](#) method.

INDEX.TS

```

import { ListView } from "@syncfusion/ej2-lists";
import { Button } from "@syncfusion/ej2-buttons";
import { MouseEventArgs } from "@syncfusion/ej2-base";
//Define an array of JSON data
let data: { [key: string]: Object }[] = [
    { text: "Hennessey Venom", id: "1", icon: "delete-icon" },
    { text: "Bugatti Chiron", id: "2", icon: "delete-icon" },
    { text: "Bugatti Veyron Super Sport", id: "3", icon: "delete-icon" },
    { text: "Aston Martin One- 77", id: "4", icon: "delete-icon" },
    { text: "Jaguar XJ220", id: "list-5", icon: "delete-icon" },
    { text: "McLaren P1", id: "6", icon: "delete-icon" }
];
//Initialize the ListView control
let listViewInstance: ListView = new ListView({
    //Set the dataSource property
    dataSource: data,
    //Map the appropriate columns to the fields property
    fields: { text: "text", iconCss: "icon" },
    //Set the template for list items
    template: "<div class='text-content'> ${text} <span class = 'delete-
icon'></span> </div>",
    //Event handler to bind the click event for delete icon
    actionComplete: onComplete
});
//Render the initialized ListView
listviewInstance.appendTo("#sample-list-flat");
//Initialize the Button control.
let button: Button = new Button();
//Render the initialized button
button.appendTo("#btn");
//Event handler to add the list item on button click
document.getElementById("btn").onclick = () => {
    let data: { [key: string]: Object } = {
        text: "Koenigsegg - " + (Math.random() * 1000).toFixed(0),
        id: (Math.random() * 1000).toFixed(0).toString(),
        icon: "delete-icon"
    };
    listViewInstance.addItem([data]);
    onComplete();
};
//Method for actionComplete handler
function onComplete() {
    let iconEle: HTMLCollection = document.getElementsByClassName("delete-
icon");
    //Event handler to bind the click event for delete icon
    Array.prototype.forEach.call(iconEle, (element: HTMLElement) => {
        element.addEventListener("click", deleteItem.bind(this));
    });
}
//Method to delete the list item
function deleteItem(args: MouseEventArgs) {
    args.stopPropagation();
    let liItem: HTMLElement = (args.target as
HTMLElement).parentElement.parentElement;
    listViewInstance.removeItem(liItem);
    onComplete();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="sample-list-flat"></div>
    <button id="btn"> Add Item </button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamic templates in listview based on device in EJ2 JavaScript Listview control

The Syncfusion Essential JS2 controls are desktop and mobile-friendly. So, you can use Syncfusion controls in both modes. The control templates are not always fixed. Applications may need to load various templates depending upon the device.

Integration

In the ListView control, template support is being used. In some cases, the control wrapper is always responsive across all devices, but the template contents are dynamically changed with unspecified (sample side) dimensions. CSS customization is also needed in sample-side to align template content responsively in both mobile and desktop modes. Here, two templates have been loaded for mobile and desktop modes. To check the device mode, a browser module has been imported from the ej2-base package.

INDEX.TS

```

import { ListView } from '@syncfusion/ej2-lists';
import { Browser } from '@syncfusion/ej2-base';
let mob_template: string = '<div class="settings">'
    + '<div id="postContainer"><div id="postImg">'
    + '<img src=${image} /></div>'
    + '<div id="content">'
    + '<div id="info">'
    + '<div id="logo"> <div id="share">'
    + '<span class="share"></span> </div> <div id="comments"> <span
class="comments"></span> </div>'
    + '<div id="bookmark"> <span class="bookmark"></span> </div></div></div>'
    + '<div class="name">${Name}</div>'
    + '<div class="description">${content}</div>'
    + '<div class="timeStamp">${timeStamp} </div>'
    + '</div>'
let template: string = '<div class="settings">'
    + '<div id="postContainer"><div id="postImg">'
    + '<img src=${image} /></div>'
    + '<div id="content">'
    + '<div class="name">${Name}</div>'
    + '<div class="description">${content}</div>'
    + '<div id="info">'
    + '<div id="logo"> <div id="share">'
    + '<span class="share"></span> </div> <div id="comments"> <span
class="comments"></span> </div>'
    + '<div id="bookmark"> <span class="bookmark"></span> </div></div>'
    + '<div class="timeStamp">${timeStamp} </div>'
    + '</div>'
    + '</div>'
//Define an array of JSON data
let dataSource: { [key: string]: Object }[] = [
    { Name: 'IBM Open-Sources Web Sphere Liberty Code', content: 'In
September, IBM announced that it would be open-sourcing the code for
WebSphere...', id: '1', image:
'https://ej2.syncfusion.com/demos/src/listview/images/1.png', timeStamp:
'Syncfusion Blog - October 19, 2017' },
    { Name: 'Must Reads: 5 Big Data E-books to upend your development',
content: 'Our first e-book was published in May 2012-jQuery Succinctly was
the start of over...', id: '2', image:
'https://ej2.syncfusion.com/demos/src/listview/images/2.png', timeStamp:
'Syncfusion Blog - October 18, 2017' },
    { Name: 'The Syncfusion Global License: Your Questions, Answered ',
content: 'Syncfusion recently hosted a webinar to cover the ins and outs of
the Syncfusion global...', id: '4', image:
'https://ej2.syncfusion.com/demos/src/listview/images/3.png', timeStamp:
'Syncfusion Blog - October 18, 2017' },
    { Name: 'Know: What is Coming from Microsoft this Fall ', content: 'On
October 17, Microsoft will release its Fall Creators Update for the Windows
10 platform...', id: '5', image:
'https://ej2.syncfusion.com/demos/src/listview/images/6.png', timeStamp:
'Syncfusion Blog - October 17, 2017' }
];
let wintemplate: string;
if (Browser.isDevice) {
    document.getElementById('List').style.width = '350px';
    wintemplate = mob_template;
}

```



```

}
else {
    wintemplate = template;
}
// Initialize the ListView control
let listObj: ListView = new ListView({
    //Set the defined data to the dataSource property
    dataSource: dataSource,
    //Map the appropriate columns to the fields property
    fields: { text: 'Name', },
    //Set customized template
    template: wintemplate,
    //Set header title
    headerTitle: 'Syncfusion Blog',
    //Set the showHeader to true to show the header title
    showHeader: true
});
//Render the initialized ListView control
listObj.appendTo('#List');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="col-lg-12 control-section">
            <!-- ListView element -->
            <div id="List" tabindex="1">

                </div>
            </div>
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Create mobile contact layout from listview in EJ2 JavaScript Listview control

You can customize the ListView using the [template](#) property. Refer to the following steps to customize ListView as mobile contact view with our [ej2-avatar](#).

- Render the ListView with [dataSource](#) that has avatar data. You can set avatar data as either text or class names. Refer to the following codes.

```
`ts
let dataSource: { [key: string]: Object }[] = [
{
text: "Jenifer", contact: "(206) 555-985774", id: "1", avatar: "", pic: "pic01"
},
{
text: "Amenda", contact: "(206) 555-3412", id: "2", avatar: "A", pic: ""
}
];
`
```

- Set [avatar](#) classes in ListView template to customize contact icon. In the following codes, medium size avatar has been set using the class name [e-avatar e-avatar-circle](#) from data source.

```
`ts
template: '<div class="e-list-wrapper e-list-multi-line e-list-avatar">' +
'${if(avatar!=="")}' +
'<span class="e-avatar e-avatar-circle">${avatar}</span>' +
'${else}' +
'<span class="${pic} e-avatar e-avatar-circle"> </span>' +
'${/if}' +
'<span class="e-list-item-header">${text}</span>' +
'<span class="e-list-content">${contact}</span>' +
'</div>';
`
```

Avatars can be set in different sizes in avatar classes. To know more about avatar classes, refer to [Avatar](#).

- Sort the contact names using the [sortOrder](#) property of ListView.
- Enable the [showHeader](#) property, and set the [headerTitle](#) as Contacts.

INDEX.TS

```
import { ListView } from "@syncfusion/ej2-lists";
let template: string = '<div class="e-list-wrapper e-list-multi-line e-list-avatar">' +
    '${if(avatar!=="")}' +
    '<span class="e-avatar e-avatar-circle">${avatar}</span>' +
    '${else}' +
    '<span class="${pic} e-avatar e-avatar-circle"> </span>' +
    '${/if}' +
    '<span class="e-list-item-header">${text}</span>' +
    '<span class="e-list-content">${contact}</span>' +
    '</div>';
//Define an array of JSON data
let dataSource: { [key: string]: Object }[] = [
    {
        text: "Jenifer",
        contact: "(206) 555-985774",
        id: "1",
        avatar: "",
        pic: "pic01"
    },
    { text: "Amenda", contact: "(206) 555-3412", id: "2", avatar: "A", pic: "" },
    {
        text: "Isabella",
        contact: "(206) 555-8122",
        id: "4",
        avatar: "",
        pic: "pic02"
    },
    {
        text: "William ",
        contact: "(206) 555-9482",
        id: "5",
        avatar: "W",
        pic: ""
    },
    {
        text: "Jacob",
        contact: "(71) 555-4848",
        id: "6",
        avatar: "",
        pic: "pic04"
    },
    { text: "Matthew", contact: "(71) 555-7773", id: "7", avatar: "M", pic: "" },
    {
        text: "Oliver",
```

```

        contact: "(71) 555-5598",
        id: "8",
        avatar: "",
        pic: "pic03"
    },
    {
        text: "Charlotte",
        contact: "(206) 555-1189",
        id: "9",
        avatar: "C",
        pic: ""
    }
];
// Initialize the ListView control
let listObj: ListView = new ListView({
    //Set the defined data to the dataSource property
    dataSource: dataSource,
    //Map the appropriate columns to the fields property
    fields: { text: "text" },
    //Set the width of the ListView
    width: "350px",
    //Enable the header of the ListView
    showHeader: true,
    //Set the header title
    headerTitle: "Contacts",
    //set cssClass for template customization
    cssClass: 'e-list-template',
    //Set the customized template
    template: template,
    sortOrder: "Ascending"
});
//Render the initialized ListView control
listObj.appendTo("#List");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="List" tabindex="1"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter and search list items using listview in EJ2 JavaScript Listview control

The filtered data can be displayed in the ListView control depending upon on user inputs using the [DataManager](#). Refer to the following steps to render the ListView with filtered data.

- Render a textbox to get input for filtering data.
- Render ListView with [dataSource](#), and set the [sortOrder](#) property.
- Bind the [keyup](#) event for textbox to perform filtering operation. To filter list data, pass the list data source to the [DataManager](#), manipulate the data using the [executeLocal](#) method, and then update filtered data as ListView dataSource.

INDEX.TS

```

import { ListView } from "@syncfusion/ej2-lists";
import { enableRipple } from "@syncfusion/ej2-base";
import { DataManager, Query } from "@syncfusion/ej2-data";
enableRipple(true);
//Define an array of JSON data
let listData: { [key: string]: Object }[] = [
    { text: "Hennessey Venom", id: "list-01" },
    { text: "Bugatti Chiron", id: "list-02" },
    { text: "Bugatti Veyron Super Sport", id: "list-03" },
    { text: "SSC Ultimate Aero", id: "list-04" },
    { text: "Koenigsegg CCR", id: "list-05" },
    { text: "McLaren F1", id: "list-06" }
];
// Initialize the ListView control
let listObj: ListView = new ListView({
    //Set the defined data to the dataSource property
    dataSource: listData,
    //Map the appropriate columns to the fields property
    fields: { text: "text", id: "id" },
    //Set the sortOrder property
    sortOrder: "Ascending"
});
//Render the initialized ListView control
listObj.appendTo("#list");
document.getElementById("textbox").addEventListener("keyup", onKeyUp);
//Here, the list items are filtered using the DataManager instance for
//ListView

```

```
function onKeyUp() {
    let value = (document.getElementById("textbox") as
HTMLInputElement).value;
    let data: Object[] = new DataManager(listData).executeLocal(
        new Query().where("text", "startswith", value, true)
    );
    if (!value) {
        listObj.dataSource = listData.slice();
    } else {
        listObj.dataSource = data as { [key: string]: Object }[];
    }
    listObj.dataBind();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="sample">
            <input class="e-input" type="text" id="textbox"
placeholder="Filter" title="Type in a name">
            <div id="list"></div>
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

In this demo, data has been filtered with starting character of the list items. You can also filter list items with ending character by passing the `endsWith` in [where](#) clause instead of `startswith`.

ListView with hyper link navigation in EJ2 JavaScript ListView control

We can use `anchor` tag along with `href` attribute in our ListView [template](#) property for navigation.

```
`ts
```

```
let anchorTemplate: string = "<a target='blank' href='${url}'>${name}</a>";
```

```
,
```

In the below sample, we have rendered `ListView` with search engines URL.

INDEX.TS

```
import { ListView } from '@syncfusion/ej2-lists';
//Define an array of JSON data
let dataSource: { [key: string]: Object }[] = [
  { name: 'Google', url: 'https://www.google.com' },
  { name: 'Bing', url: 'https://www.bing.com' },
  { name: 'Yahoo', url: 'https://www.yahoo.com' },
  { name: 'Ask.com', url: 'https://www.ask.com' },
  { name: 'AOL.com', url: 'https://www.aol.com' }
];
let anchor_template: string = "<a target='_blank'
href='${url}'>${name}</a>";
// Initialize ListView control
let listObj: ListView = new ListView({
  //Set defined data to dataSource property
  dataSource: dataSource,
  //Set header title
  headerTitle: 'Search engines',
  //Set true to show header title
  showHeader: true,
  template: anchor_template
});
//Render initialized ListView control
listObj.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Chat window user interface using listview in EJ2 JavaScript Listview control

ListView can be customized as chat window. To achieve that, use the ListView [template](#) property and [Avatar](#) control.

- The ListView template property is used to showcase the ListView as chat window.
- Avatar control is used to design the image of contact person.

Refer the below template code snippet for Template of chat window.

```

`ts
let template =
'<div class="settings">' +
'${if(chat!="receiver")}' +
'<div id="content">' +
'<div class="name">${text}</div>' +
'<div id="info">${contact}</div></div>' +
'${if(avatar!="")}' +
'<div id="image"><span class="e-avatar img1 e-avatar-circle">${avatar}</span></div>' +
"${else}" +
'<div id="image"><span class="${pic} img1 e-avatar e-avatar-circle"> </span></div>' +
"${/if}" +
"${else}" +

```



```
'${if(avatar!="")}' +
'<div id="image2"><span class="e-avatar img2 e-avatar-circle">${avatar}</span></div>' +
"${else}" +
'<div id="image2"><span class="${pic} img2 e-avatar e-avatar-circle"> </span></div>' +
"${if}" +
'<div id="content1">' +
'<div class="name1">${text}</div>' +
'<div id="info1">${contact}</div>' +
"</div>" +
"${if}" +
"</div>";
`
```

Chat order in template

In ListView template, we have rendered the list items based on receiver and sender information from dataSource of listview.

Adding messages to chat window

- Use textbox to get message from user.
- Add the textbox message to ListView dataSource using [addItem](#) method.

```
`ts
document.getElementById("btn").addEventListener("click", e => {
var value = document.getElementById("name").value;
document
.getElementById("List")
.ej2_instances[0].addItem([
{
text: "Amenda",
contact: value,
id: "2",
avatar: "A",
pic: "",
chat: "receiver"
}
]);
`
```

```
});
```

```
,
```

INDEX.TS

```
import { ListView } from "@syncfusion/ej2-lists";
import { Button } from "@syncfusion/ej2-buttons";
let template: any =
    '<div class="settings">' +
    '${if(chat!=="receiver")}' +
    '<div id="content">' +
    '<div class="name">${text}</div>' +
    '<div id="info">${contact}</div></div>' +
    '${if(avatar!=="")}' +
    '<div id="image"><span class="e-avatar img1 e-avatar-circle">${avatar}</span></div>' +
    '${else}' +
    '<div id="image"><span class="${pic} img1 e-avatar e-avatar-circle">
</span></div>' +
    '${/if}' +
    '${else}' +
    '${if(avatar!=="")}' +
    '<div id="image2"><span class="e-avatar img2 e-avatar-circle">${avatar}</span></div>' +
    '${else}' +
    '<div id="image2"><span class="${pic} img2 e-avatar e-avatar-circle">
</span></div>' +
    '${/if}' +
    '<div id="content1">' +
    '<div class="name1">${text}</div>' +
    '<div id="info1">${contact}</div>' +
    '</div>' +
    '${/if}' +
    '</div>';
//Define an array of JSON data
let dataSource: any = [
    {
        text: "Jenifer",
        contact: "Hi",
        id: "1",
        avatar: "",
        pic: "pic01", chat: "sender"
    },
    { text: "Amenda", contact: "Hello", id: "2", avatar: "A", pic: "", chat:
"receiver" },
    {
        text: "Jenifer",
        contact: "What Knid of application going to launch",
        id: "4",
        avatar: "",
        pic: "pic02", chat: "sender"
    },
    {
        text: "Amenda ",
        contact: "A knid of Emergency broadcast App",
        id: "5",
```

```

        avatar: "A",
        pic: "", chat: "receiver"
    },
    {
        text: "Jacob",
        contact: "Can you please elaborate",
        id: "6",
        avatar: "",
        pic: "pic04", chat: "sender"
    },
];
// Initialize the ListView component
let listObj: ListView = new ListView({
    //Set the defined data to the dataSource property
    dataSource: dataSource,
    //Map the appropriate columns to the fields property
    fields: { text: "Name" },
    //Set the width of the ListView
    width: "350px",
    //Enable the header of the ListView
    showHeader: true,
    //Set the header title
    headerTitle: "Chat",
    //Set the customized template
    template: template,
});
//Render the initialized ListView component
listObj.appendTo("#List");
let button: Button = new Button();
// Render initialized button.
button.appendTo('#btn');
document.getElementById('btn').addEventListener('click', (e) => {
    let value = (document.getElementById('name') as HTMLElement).value;
    listObj.addItem([
        { text: "Amenda", contact: value, id: "2", avatar: "A",
        pic: "", chat: "receiver" }
    ]);
    (document.getElementById('name') as HTMLElement).value = "";
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="List" tabindex="1"></div>
        <div style="width: 350px;margin: 0 auto;"><input id="name"
style="width: 275px" class="e-input" type="text" placeholder="Type your
message">
        <button id="btn" style="float:right">Send</button></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop list items in listview in EJ2 JavaScript Listview control

In ListView control, we don't have drag and drop support. But we can achieve this requirement using [TreeView](#) control with ListView appearance.

Drag and Drop in TreeView control was enabled by setting [allowDragAndDrop](#) to `true`.

```
`ts
```

```
let treeViewInstance: TreeView = new TreeView({
fields: { dataSource: data, id: 'id', text: 'text' },
allowDragAndDrop: true
```

```
});
```

```
,
```

The TreeView control is used to represent hierarchical data in a tree like structure. So, list items in TreeView can be dropped to child of target element. we can prevent this behaviour by cancelling the [nodeDragStop](#) and [nodeDragging](#) events.

```
`ts
```

```
let treeViewInstance: TreeView = new TreeView({
fields: { dataSource: data, id: 'id', text: 'text' },
allowDragAndDrop: true,
nodeDragging: onDragStop,
```

```

nodeDragStop: onDragStop
});
function onDragStop(args: DragAndDropEventArgs) {
//Block the Child Drop operation in TreeView
let draggingItem: HTMLCollection = document.getElementsByClassName("e-drop-in");
if (draggingItem.length == 1) {
draggingItem[0].classList.add('e-no-drop');
args.cancel = true;
}
}

```

In the below sample, we have rendered draggable list items.

INDEX.TS

```

import { TreeView, DragAndDropEventArgs } from '@syncfusion/ej2-
navigations';
let data: { [key: string]: Object }[] = [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
let treeViewInstance: TreeView = new TreeView({
    fields: { dataSource: data, id: 'id', text: 'text' },
    allowDragAndDrop: true,
    nodeDragging: onDragStop,
    nodeDragStop: onDragStop
});
treeViewInstance.appendTo('#element');
function onDragStop(args: DragAndDropEventArgs) {
    //Block the Child Drop operation in TreeView
    let draggingItem: HTMLCollection = document.getElementsByClassName("e-
drop-in");
    if (draggingItem.length == 1) {
        draggingItem[0].classList.add('e-no-drop');
        args.cancel = true;
    }
}

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 for ListView </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for ListView UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Create dual list from listview in EJ2 JavaScript Listview control

The dual list contains two ListView. This allows you to move list items from one list to another using the client-side events. This section explains how to integrate the ListView control to achieve dual list.

Use cases

- Stock exchanges of two different countries
- Job applications (skill sets)

Integration of Dual List

Here, two ListView controls have been used to display the list items. An ej2-button is used to transfer data between the ListView, and a textbox is used to achieve the UI of filtering support.

The dual list supports:

- Moving whole data from one list to another.
- Moving selected data from one list to another.
- Filtering the list by using a client-side typed character.

In the ListView control, sorting is enabled using the [sortOrder](#) property, and the [select](#) event is triggered while selecting an item. Here, the select event is triggered to enable and disable button states.

Manipulating data

Moving whole data from the first list to the second list(>>)

Here, the whole data can be moved from the first ListView to the second by clicking the first button. When clicking the button, the whole list items are sliced, and `concat` with the second ListView. This button is enabled only when the data source of the first ListView is not empty.

Moving whole data from the second list to the first list(<<)**

The functionality of the second button is the same as above, and data is transferred from the second list to the first list. This button is enabled only when the data source of the second ListView is not empty.

Moving selected item from one list to another list (>) and (<)**

The [Select](#) event is triggered when selecting a list item in the ListView. The selected items can be transferred between two lists. These buttons will be enabled when selecting an item in lists.

Filtering method

The filtering method is used to filter list items when typing a character in the text box. In this method, the [dataManager](#) has been used to fetch data from the data source and display in ListView.

Sorting

By using the dual list, list items can be sorted in the ListView control using the [sortOrder](#) property.

You can enable sorting in one ListView; in the same order, data can be transferred to another ListView.

INDEX.TS

```
import { ListView, SelectedItem } from '@syncfusion/ej2-lists';
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
import { DataManager, Query } from '@syncfusion/ej2-data';
enableRipple(true);
//Define an array of JSON data
let firstListData: { [key: string]: Object }[] = [
  { text: 'Hennessey Venom', id: 'list-01' },
  { text: 'Bugatti Chiron', id: 'list-02' },
  { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
  { text: 'SSC Ultimate Aero', id: 'list-04' },
  { text: 'Koenigsegg CCR', id: 'list-05' },
  { text: 'McLaren F1', id: 'list-06' },
];
let secondListData: { [key: string]: Object }[] = [
  { text: 'Aston Martin One- 77', id: 'list-07' },
  { text: 'Jaguar XJ220', id: 'list-08' },
  { text: 'McLaren P1', id: 'list-09' },
  { text: 'Ferrari LaFerrari', id: 'list-10' },
];
// Initialize the ListView control
let listObj1: ListView = new ListView({
  //Set the defined data to the dataSource property
  dataSource: firstListData.slice(),
  //Map the appropriate columns to the fields property
  fields: { text: 'text', id: 'id' },
  sortOrder: 'Ascending',
  select: onFirstListSelect
});
//Render the initialized ListView control
listObj1.appendTo('#list-1');
```

```

let listObj2: ListView = new ListView({
    //Set the defined data to the dataSource property
    dataSource: secondListData.slice(),
    //Map the appropriate columns to the fields property
    fields: { text: 'text', id: 'id' },
    sortOrder: 'Ascending',
    select: onSeconListSelect
});
//Render the initialized ListView control
listObj2.appendTo('#list-2');
let btnobj1: Button = new Button();
btnobj1.appendTo('#firstBtn');
let btnobj2: Button = new Button({
    disabled: true
});
btnobj2.appendTo('#secondBtn');
let btnobj3: Button = new Button({
    disabled: true
});
btnobj3.appendTo('#thirdBtn');
let btnobj4: Button = new Button();
btnobj4.appendTo('#fourthBtn');
//Here, all list items are moved to the second list on clicking move all
button
btnobj1.element.addEventListener('click', () => {
    listObj2.dataSource = Array.prototype.concat.call(listObj1.dataSource,
listObj2.dataSource);
    listObj2.dataBind();
    updateFirstListData();

listObj1.removeMultipleItems(Array.prototype.slice.call(listObj1.element.que
rySelectorAll('.e-list-item')));
    firstListData = firstListData.concat(listObj1.dataSource as { [key:
string]: Object }[]);
    secondListData = (listObj2.dataSource as { [key: string]: Object
}[]).slice();
    btnobj1.disabled = true;
    onFirstKeyUp();
    setButtonState();
});
//Here, the selected list items are moved to the second list on clicking
move button
btnobj2.element.addEventListener('click', () => {
    let selectedItem: SelectedItem = listObj1.getSelectedItems() as
SelectedItem;
    listObj2.dataSource = Array.prototype.concat.call(listObj2.dataSource,
selectedItem.data);
    listObj2.dataBind();
    updateFirstListData();
    listObj1.removeItem(selectedItem.item);
    firstListData = firstListData.concat(listObj1.dataSource as { [key:
string]: Object }[]);
    secondListData = (listObj2.dataSource as { [key: string]: Object
}[]).slice();
    onFirstKeyUp();
    btnobj2.disabled = true;
    setButtonState();

```



```

});
//Here, the selected list items are moved to the first list on clicking move button
btnobj3.element.addEventListener('click', () => {
    let selectedItem: SelectedItem = listObj2.getSelectedItems() as
SelectedItem;
    listObj1.dataSource = Array.prototype.concat.call(listObj1.dataSource,
selectedItem.data);
    listObj1.dataBind();
    updateSecondListData();
    listObj2.removeItem(selectedItem.item);
    secondListData = secondListData.concat(listObj2.dataSource as { [key:
string]: Object }[]);
    firstListData = (listObj1.dataSource as { [key: string]: Object
}[]).slice();
    onSecondKeyUp();
    btnobj3.disabled = true;
    setButtonState();
});
//Here, all list items are moved to the first list on clicking move all button
btnobj4.element.addEventListener('click', () => {
    listObj1.dataSource = Array.prototype.concat.call(listObj1.dataSource,
listObj2.dataSource);
    listObj1.dataBind();
    updateSecondListData();

listObj2.removeMultipleItems(Array.prototype.slice.call(listObj2.element.que
rySelectorAll('.e-list-item')));
    secondListData = secondListData.concat(listObj2.dataSource as { [key:
string]: Object }[]);
    firstListData = (listObj1.dataSource as { [key: string]: Object
}[]).slice();
    onSecondKeyUp();
    setButtonState();
});
//Here, the ListView data source is updated to the first list
function updateFirstListData() {
    Array.prototype.forEach.call(listObj1.element.querySelectorAll('.e-list-
item'), (list: HTMLLIElement) => {
        firstListData.forEach((data, index) => {
            if (list.innerText.trim() === data.text) {
                firstListData.splice(index, 1)
            }
        });
    });
    (document.getElementById("firstInput") as HTMLInputElement).value = '';
    let ds: { [key: string]: Object }[] = [];
    firstListData.forEach((data) => {
        ds.push(data);
    })
    firstListData = ds;
}
//Here, the ListView dataSource is updated for the second list
function updateSecondListData() {
    Array.prototype.forEach.call(listObj2.element.querySelectorAll('.e-list-
item'), (list: HTMLLIElement) => {

```

```

        secondListData.forEach((data, index) => {
            if (list.innerText.trim() === data.text) {
                secondListData.splice(index, 1)
            }
        });
    });
    (document.getElementById("secondInput" as HTMLInputElement).value = '');
    let ds: { [key: string]: Object }[] = [];
    secondListData.forEach((data) => {
        ds.push(data);
    })
    secondListData = ds;
}
function onFirstListSelect() {
    btnobj2.disabled = false;
}
function onSeconListSelect() {
    btnobj3.disabled = false;
}
document.getElementById('firstInput').addEventListener('keyup',
onFirstKeyUp);
//Here, filtering is handled using the DataManager for the first list
function onFirstKeyUp() {
    let value: string = (document.getElementById("firstInput" as
HTMLInputElement).value;
    var data: Object[] = new DataManager(firstListData).executeLocal(new
Query().where('text', 'startswith', value, true));
    if (!value) {
        listObj1.dataSource = firstListData.slice();
    } else {
        listObj1.dataSource = data as { [key: string]: Object }[];
    }
    listObj1.dataBind();
}
document.getElementById('secondInput').addEventListener('keyup',
onSecondKeyUp);
//Here, filtering is handled using the DataManager for the second list
function onSecondKeyUp() {
    let value: string = (document.getElementById("secondInput" as
HTMLInputElement).value;
    var data: Object[] = new DataManager(secondListData).executeLocal(new
Query().where('text', 'startswith', value, true));
    if (!value) {
        listObj2.dataSource = secondListData.slice();
    } else {
        listObj2.dataSource = data as { [key: string]: Object }[];
    }
    listObj2.dataBind();
}
//Here, the state of the button is changed
function setButtonState() {
    if ((listObj1.dataSource as { [key: string]: Object }[]).length) {
        btnobj1.disabled = false;
    } else {
        btnobj1.disabled = true;
        btnobj2.disabled = true;
    }
}

```

```

    if ((listObj2.dataSource as { [key: string]: Object }[]).length) {
        btnobj4.disabled = false;
    } else {
        btnobj4.disabled = true;
        btnobj3.disabled = true;
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <h3>Dual List</h3>
        <div class="list_container">
            <div id="list_container_1">
                <input class="e-input" type="text" id="firstInput"
placeholder="Filter" title="Type in a name">
                <div id="list-1"></div>
            </div>
            <div id="btn">
                <button id="firstBtn"> &#62;&#62; </button>
                <button id="secondBtn"> &#62; </button>
                <button id="thirdBtn">
                    &#60; </button>
                <button id="fourthBtn">
                    &#60;&#60; </button>
            </div>
            <div id="list_container_2">
                <input class="e-input" type="text" id="secondInput"
placeholder="Filter" title="Type in a name">
                <div id="list-2"></div>
            </div>

```

```

        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamic listview by loading ajax html content in EJ2 JavaScript Listview control

We can set external **HTML** page content as [template](#) for our **ListView** control by making use of **AJAX** call.

```

`ts
let template: string;
let ajax: Ajax = new Ajax('./template.html', 'GET', false);
ajax.onSuccess = (e: string)=>{
    template = e;
}
ajax.send();
`

```

In the below sample, we have rendered smartphone settings template from external **HTML** file.

INDEX.TS

```

import { ListView } from '@syncfusion/ej2-lists';
import { Ajax } from '@syncfusion/ej2-base';
let data: { [key: string]: Object }[] = [
    { name: 'Network & Internet', id: '0', description: 'Wi-Fi, mobile, data usage, hotspot' },
    { name: 'Connected devices', id: '1', description: 'Bluetooth, cast, NFC' },
    { name: 'Battery', id: '2', description: '18% -4h 12m left' },
    { name: 'Display', id: '3', description: 'Wallpaper, sleep, font size' },
    { name: 'Sound', id: '4', description: 'Volume, vibration, Do Not Disturb' },
    { name: 'Storage', id: '5', description: '52% used - 15.48 GB free' }
];
let template: string;
let ajax: Ajax = new Ajax('./template.html', 'GET', false);
ajax.onSuccess = (e: string)=>{
    template = e;
}
ajax.send();
let listViewInstance: ListView = new ListView({
    dataSource: data,
    template: template,

```

```

        headerTitle: 'Settings',
        showHeader: true,
        fields: {text:'name', id: 'id'}
    });
    listViewInstance.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Display spinner for list item loading from remote data in EJ2 JavaScript Listview control

The features of the ListView control such as remote data-binding take more time to fetch data from corresponding dataSource/remote URL. In this case, you can use EJ2 [Spinner](#) to enhance the appearance of the UI. This section explains how to load a spinner control to groom the appearance.

Refer to the following code sample to render the spinner control.

```

`ts
createSpinner({
target: document.getElementById('spinner')
});
showSpinner(document.getElementById('spinner'));

```

Refer to the following code sample to render the ListView control.

```
`ts
let listViewInstance: ListView = new ListView({
//Bind the DataManager instance to the dataSource property
dataSource: new DataManager({
url: '///js.syncfusion.com/ejServices/Wcf/Northwind.svc/',
crossDomain: true
}),
//Bind the Query instance to the query property
query: new Query().from('Products').select('ProductID,ProductName').take(10),
//Map the appropriate columns to the fields property
fields: { id: 'ProductID', text: 'ProductName' },
//Set the header tittle to the list
headerTitle: 'Product Name',
showHeader: true,
width:"300",
actionComplete : oncomplete
});
//Render the initialized ListView
listviewInstance.appendTo("#element");
```

Here, the data is fetched from **Northwind** Service URL; it takes a few seconds to load the data. To enhance the UI, the spinner control has been rendered initially. After the data is loaded from remote URL, the spinner control will be hidden in ListView [actionComplete](#) event.

INDEX.TS

```
import { ListView } from '@syncfusion/ej2-lists';
//Import DataManager related classes
import { DataManager, Query } from '@syncfusion/ej2-data';
import { createSpinner, showSpinner } from '@syncfusion/ej2-popups';
//Initialize the ListView control
let listViewInstance: ListView = new ListView({
    //Bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.syncfusion.com/js/production/api/',
        crossDomain: true
    }),
    //Initialize query with the Query instance to get specified set of data
```

```

    query: new
    Query().from('ListView').select('EmployeeID,FirstName').take(10),
    //Map the appropriate columns to fields property
    fields: { id: 'EmployeeID', text: 'FirstName' },
    //Set header title
    headerTitle: 'Employees',
    //Set true to show header title
    showHeader: true,
    width:"300",
    actionComplete : oncomplete
});
//Render the initialized ListView
listviewInstance.appendTo("#element");
createSpinner({
    target: document.getElementById('spinner')
});
showSpinner(document.getElementById('spinner'));
function oncomplete() {
    document.getElementById('spinner').style.display = "none";
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element">
            <div id="spinner"></div>
        </div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamic hierarchical listview in EJ2 JavaScript Listview control

To dynamic hierarchical listview, push the new list item data into the existing [dataSource](#) using the [select](#) event.

Refer to the following steps to load list item into the child list:

1. Initially, render the ListView with the required data source.
2. Bind the [select](#) event that triggers selecting list item in the ListView control. By using the select event, you can push the new list item to the child list of the data source on specifying its item index. Item index can be obtained from the

[SelectEventArgs](#) of the select event.

INDEX.TS

```

import { ListView, SelectEventArgs } from '@syncfusion/ej2-lists';
let dataSource: { [key: string]: Object }[] = [
  {
    id: '01', text: 'Music', icon: 'folder',
    child: [
      { id: '01-01', text: 'Gouttes.mp3', icon: 'file' }
    ]
  },
  {
    id: '02', text: 'Videos', icon: 'folder',
    child: [
      { id: '02-01', text: 'Naturals.mp4', icon: 'file' },
      { id: '02-02', text: 'Wild.mpeg', icon: 'file' },
    ]
  },
  {
    id: '03', text: 'Documents', icon: 'folder',
    child: [
      { id: '03-01', text: 'Environment Pollution.docx', icon: 'file' },
      { id: '03-02', text: 'Global Water, Sanitation, & Hygiene.docx', icon: 'file' },
      { id: '03-03', text: 'Global Warming.ppt', icon: 'file' },
      { id: '03-04', text: 'Social Network.pdf', icon: 'file' },
      { id: '03-05', text: 'Youth Empowerment.pdf', icon: 'file' },
    ]
  },
  {
    id: '04', text: 'Pictures', icon: 'folder',
    child: [
      {
        id: '04-01', text: 'Camera Roll', icon: 'folder',
        child: [
          { id: '04-01-01', text: 'WIN_20160726_094117.JPG', icon: 'file' },
        ]
      }
    ]
  }
]

```



```

        { id: '04-01-02', text: 'WIN_20160726_094118.JPG', icon:
'file' },,
        { id: '04-01-03', text: 'WIN_20160726_094119.JPG', icon:
'file' }
    ]
},
{
    id: '04-02', text: 'Wind.jpg', icon: 'file'
},
{
    id: '04-02', text: 'Stone.jpg', icon: 'file'
},
{
    id: '04-02', text: 'Home.jpg', icon: 'file'
},
{
    id: '04-02', text: 'Bridge.png', icon: 'file'
}
]
},
{
    id: '05', text: 'Downloads', icon: 'folder',
    child: [
        { id: '05-01', text: 'UI-Guide.pdf', icon: 'file' },
        { id: '05-02', text: 'Tutorials.zip', icon: 'file' },
        { id: '05-03', text: 'Game.exe', icon: 'file' },
        { id: '05-04', text: 'TypeScript.7z', icon: 'file' },
    ]
},
];
//Initialize the listview control
let listViewInstance: ListView = new ListView({
    //Set the defined data to the dataSource property
    dataSource: dataSource,
    //Map the appropriate columns to the fields property
    fields: { iconCss: 'icon', tooltip: 'text' },
    //Set the showIcon to true
    showIcon: true,
    //Set header title
    headerTitle: 'Folders',
    //Set the showHeader to true
    showHeader: true,
    //Select onSelect event to add new list item in child page
    select: onSelect
});
//Render the initialized ListView
listviewInstance.appendTo("#listview");
//Select event to add new list item in child page
function onSelect(args: SelectEventArgs) {
    //Add new file to the child page of selected list item
    this.dataSource[args.index].child.push({ id: '01-02', text: 'Newly Added
File', icon: 'file', htmlAttributes: { role: 'li', class: 'list' } });
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="listview"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Hide checkbox in listview in EJ2 JavaScript Listview control

The checkbox of the any list item can be hidden by using [htmlAttributes](#) of [fields](#) object. With the help of [htmlAttributes](#) we can add unique class to each list item that will be rendered from the data source, from the CSS class we can hide the checkbox of the list item.

In this sample, we had hidden the multiple leaf node of nested list. The `e-checkbox-hidden` class has been added in the data source where the checkbox needs to be hidden. Refer the below snippet for simple data source.

```

`ts
{
  'text': 'New York',
  'id': '3002',
  'category': 'USA',
  'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' }
}
,

```

Even though we have hidden the checkbox the functionality will be same for the list item which might affect the `getSelectedItems` method. So, to counteract that we will follow certain logic in the `select` event. The Logic here is to remove the `e-active` class from the other checkbox hidden list item which will be added when we select on that item and retain `e-active` on currently selected item.

In this process we will exclude the visible checkbox list items and only consider the hidden checkbox items.

INDEX.TS

```
import { ListView, SelectEventArgs } from '@syncfusion/ej2-lists';
//define the array of JSON
let dataSource: { [key: string]: Object }[] = [
  {
    'text': 'Asia',
    'id': '01',
    'category': 'Continent',
    'child': [{
      'text': 'India',
      'id': '1',
      'category': 'Asia',
      'child': [{
        'text': 'Delhi',
        'id': '1001',
        'category': 'India',
        'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
      },
      {
        'text': 'Kashmir',
        'id': '1002',
        'category': 'India',
        'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
      },
      {
        'text': 'Goa',
        'id': '1003',
        'category': 'India',
        'htmlAttributes': { 'class': 'e-file' },
      },
    ]
  },
  {
    'text': 'China',
    'id': '2',
    'category': 'Asia',
    'child': [{
      'text': 'Zhejiang',
      'id': '2001',
      'category': 'China',
      'htmlAttributes': { 'class': 'e-file' },
    },
    {
      'text': 'Hunan',
      'id': '2002',
      'category': 'China',
      'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
    },
  ],
}
```

```

    },
    {
        'text': 'Shandong',
        'id': '2003',
        'category': 'China',
        'htmlAttributes': { 'class': 'e-file' },
    }
  ]
},
{
  'text': 'North America',
  'id': '02',
  'category': 'Continent',
  'child': [{
    'text': 'USA',
    'id': '3',
    'category': 'North America',
    'child': [{
      'text': 'California',
      'id': '3001',
      'category': 'USA',
      'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
    },
    {
      'text': 'New York',
      'id': '3002',
      'category': 'USA',
      'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
    },
    {
      'text': 'Florida',
      'id': '3003',
      'category': 'USA',
      'htmlAttributes': { 'class': 'e-file' },
    }
  ]
},
{
  'text': 'Canada',
  'id': '4',
  'category': 'North America',
  'child': [{
    'text': 'Ontario',
    'id': '4001',
    'category': 'Canada',
    'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
  },
  {
    'text': 'Alberta',
    'id': '4002',
    'category': 'Canada',
    'htmlAttributes': { 'class': 'e-file' },
  },
  {
    'text': 'Manitoba',
    'id': '4003',
    'category': 'Canada',
    'htmlAttributes': { 'class': 'e-file' },
  }
}

```

```

    ]]
  },
  {
    'text': 'Europe',
    'id': '03',
    'category': 'Continent',
    'child': [{
      'text': 'Germany',
      'id': '5',
      'category': 'Europe',
      'child': [{
        'text': 'Berlin',
        'id': '5001',
        'category': 'Germany',
        'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
      },
      {
        'text': 'Bavaria',
        'id': '5002',
        'category': 'Germany',
        'htmlAttributes': { 'class': 'e-file' },
      },
      {
        'text': 'Hesse',
        'id': '5003',
        'category': 'Germany',
        'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
      }
    ]
  }, {
    'text': 'France',
    'id': '6',
    'category': 'Europe',
    'child': [{
      'text': 'Paris',
      'id': '6001',
      'category': 'France',
      'htmlAttributes': { 'class': 'e-file' },
    },
    {
      'text': 'Lyon',
      'id': '6002',
      'category': 'France',
      'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
    },
    {
      'text': 'Marseille',
      'id': '6003',
      'category': 'France',
      'htmlAttributes': { 'class': 'e-file' },
    }
  ]
  },
  {
    'text': 'Australia',
    'id': '04',
    'category': 'Continent',

```

```

        'child': [{
            'text': 'Australia',
            'id': '7',
            'category': 'Australia',
            'child': [{
                'text': 'Sydney',
                'id': '7001',
                'category': 'Australia',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
                'text': 'Melbourne',
                'id': '7002',
                'category': 'Australia',
                'htmlAttributes': { 'class': 'e-file' },
            },
            {
                'text': 'Brisbane',
                'id': '7003',
                'category': 'Australia',
                'htmlAttributes': { 'class': 'e-file' },
            }
        ]
    }, {
        'text': 'New Zealand',
        'id': '8',
        'category': 'Australia',
        'child': [{
            'text': 'Milford Sound',
            'id': '8001',
            'category': 'New Zealand',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Tongariro National Park',
            'id': '8002',
            'category': 'New Zealand',
            'htmlAttributes': { 'class': 'e-file' },
        },
        {
            'text': 'Fiordland National Park',
            'id': '8003',
            'category': 'New Zealand',
            'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
        }
    ]
}
], {
    'text': 'Africa',
    'id': '05',
    'category': 'Continent',
    'child': [{
        'text': 'Morocco',
        'id': '9',
        'category': 'Africa',
        'child': [{
            'text': 'Rabat',
            'id': '9001',

```

```

        'category': 'Morocco',
        'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
    },
    {
        'text': 'Toubkal',
        'id': '9002',
        'category': 'Morocco',
        'htmlAttributes': { 'class': 'e-file' },
    },
    {
        'text': 'Todgha Gorge',
        'id': '9003',
        'category': 'Morocco',
        'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
    }
  ]],
  {
    'text': 'South Africa',
    'id': '10',
    'category': 'Africa',
    'child': [{
      'text': 'Cape Town',
      'id': '10001',
      'category': 'South Africa',
      'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
    },
    {
      'text': 'Pretoria',
      'id': '10002',
      'category': 'South Africa',
      'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
    },
    {
      'text': 'Bloemfontein',
      'id': '10003',
      'category': 'South Africa',
      'htmlAttributes': { 'class': 'e-file' },
    }
  ]
}
];

//Initialize ListView control
let listViewInstance: ListView = new ListView({
  //set the data to datasource property
  dataSource: dataSource,
  // map the groupBy field with category column
  fields: { tooltip: 'text' },
  headerTitle: 'Mixed Leaf Checkbox Hidden List ',
  showHeader: true,
  showCheckBox: true,
  select: (args: SelectEventArgs) => {
    // Selecting all the e-active elements from the list.
    let normalElements: HTMLElement[] =
Array.prototype.slice.call(listViewInstance.element.getElementsByClassName('
e-checkbox-hidden'));
    // Looping through all the selected element and removing e-active
class
    // to avoid behaviour interference with getSelectedItems method

```

```

        normalElements.forEach((element) => {
            element.classList.remove('e-active');
        });
        // Finally adding e-active class to currently selected item except
checkbox item.
        // because if it is checkbox item their actions will taken care from
the source side itself.
        if (args.item.classList.contains('e-checkbox-hidden')) {
            args.item.classList.add('e-active');
        }
    }
});
//Render initialized ListView
listviewInstance.appendTo("#folderCheckbox");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="sample">
            <!-- ListView element -->
            <div id="folderCheckbox" tabindex="1"></div>
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```


Display items count in group header in EJ2 JavaScript Listview control

The ListView control supports wrapping list items into a group based on the category. The category of each list item can be mapped with groupBy field of the data source. You can display grouped list items count in the list-header using the group header template. Refer to the following code sample to display grouped list item count.

INDEX.TS

```
import { ListView } from '@syncfusion/ej2-lists';
let template: string = '<div class="e-list-wrapper e-list-multi-line e-list-avatar">' +
    '<img class="e-avatar e-avatar-circle" src=${image} ' +
    'style="background:#BCBCBC" />' +
    '<span class="e-list-item-header">${Name}</span>' +
    '<span class="e-list-content">${contact}</span>' +
    '</div>';
//Define an array of JSON data
let dataSource: { [key: string]: string }[] = [
    { Name: 'Nancy', contact: '(206) 555-985774', id: '1', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/1.png', category:
    'Experience' },
    { Name: 'Janet', contact: '(206) 555-3412', id: '2', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/3.png', category:
    'Fresher' },
    { Name: 'Margaret', contact: '(206) 555-8122', id: '4', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/4.png', category:
    'Experience' },
    { Name: 'Andrew ', contact: '(206) 555-9482', id: '5', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/2.png', category:
    'Experience' },
    { Name: 'Steven', contact: '(71) 555-4848', id: '6', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/5.png', category:
    'Fresher' },
    { Name: 'Michael', contact: '(71) 555-7773', id: '7', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/6.png', category:
    'Experience' },
    { Name: 'Robert', contact: '(71) 555-5598', id: '8', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/7.png', category:
    'Fresher' },
    { Name: 'Laura', contact: '(206) 555-1189', id: '9', image:
    'https://ej2.syncfusion.com/demos/src/grid/images/8.png', category:
    'Experience' },
];
// Initialize the ListView control
let listObj: ListView = new ListView({
    //Set the defined data to the data source property
    dataSource: dataSource,
    //Map the appropriate columns to the fields property
    fields: { text: 'Name', groupBy: 'category' },
    //set cssClass for template customization
    cssClass: 'e-list-template',
    //Set the customized template
    template: template,
    groupTemplate: '<div><span class="category">${items[0].category}</span>
    <span class="count"> ${items.length} Item(s)</span></div> '
```

```
});
//Render the initialized ListView control
listObj.appendTo('#List');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="col-lg-12 control-section">
      <!-- ListView element -->
      <div id="List" tabindex="1">
        </div>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customize listview as grid layout in EJ2 JavaScript Listview control

In ListView, list items can be rendered in grid layout with following data manipulations.

- Add Item
- Remove Item
- Sort Items
- Filter Items

Grid Layout

In this section, we will discuss about rendering of list items in grid layout.

- Initialize and render ListView with dataSource which will render list items in list layout.
- Now, add the below CSS to list item. This will make list items to render in grid layout

```

default-list .e-list-item {
height: 100px;

width: 100px;

float: left;
}

```

In the below sample, we have rendered List items in grid layout.

INDEX.TS

```

import { ListView } from '@syncfusion/ej2-lists';
//define the array of string
let data: number[] = [1, 2, 3, 4, 5, 6, 7, 8, 9];
//Initialize ListView control
let listViewInstance: ListView = new ListView({
    //set the data to datasource property
    dataSource: data,
    //set the template for list items
    template: ''
});
//Render initialized ListView
listViewInstance.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>

  <div id="container">
    <div id="element"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Data manipulation

In this section, we will discuss about ListView data manipulations.

Add Item

We can add list item using [addItem](#) API. This will accept array of data as argument.

```
`ts
```

```
listViewInstance.addItem([{'text': 'Apricot', id: '32'}]);
```

```
,
```

In the below sample, you can add new fruit item by clicking add button which will open dialog box with fruit name and image URL text box. After entering the item details, click the add button. This will add your new fruit item.

Remove item

We can remove list item using [removeItem](#) API. This will accept fields with `id` or list item element as argument.

```
`ts
```

```
listViewInstance.removeItem({id: '32'});
```

```
,
```

In the below sample, you can remove fruit by hovering the fruit item which will show delete button and click that delete button to delete that fruit from your list.

Sort Items

ListView can be sorted either in Ascending or Descending order. To enable sorting in your ListView, set [sortOrder](#) as `Ascending` or `Descending`.

```
`ts
```

```
let listViewInstance: ListView = new ListView({
  sortOrder: 'Ascending';
})
```

```
,
```

We can also set sorting after control initialization.

```
`ts
```

```
listViewInstance.sortOrder = 'Ascending'
```

```
,
```

In the below sample, we have sorted fruits in **Ascending** order. To sort it in descending, click on sort order icon and vice versa.

Filter Items

Listview data can be filtered with the help of [dataManager](#). After filtering the data, update ListView [dataSource](#) with filtered data.

```
`ts
```

```
let value = document.getElementById("filter").value; //input text box value
```

```
let filteredData = new DataManager(listdata).executeLocal(
```

```
new Query().where("text", "startswith", value, true)
```

```
);
```

```
listViewInstance.dataSource = filteredData;
```

```
,
```

In the below sample, we can filter fruit items with the help of search text box. This will filter fruit items based on your input. Here we used [startswith](#) of input text to filter data in DataManager.

INDEX.TS

```
import { ListView, SelectedItem } from '@syncfusion/ej2-lists';
import { Dialog } from '@syncfusion/ej2-popups';
import { closest, enableRipple, MouseEventArgs } from '@syncfusion/ej2-base';
import { DataManager, Query } from '@syncfusion/ej2-data';
enableRipple(true);
let ascClass = 'e-sort-icon-ascending';
let desClass = 'e-sort-icon-descending';
//define the array of JSON
let fruitsdata: { [key: string]: Object }[] = [
  { text: 'Date', id: '1', imgUrl: './dates.jpg' },
  { text: 'Fig', id: '2', imgUrl: './fig.jpg' },
  { text: 'Apple', id: '3', imgUrl: './apple.png' },
  { text: 'Apricot', id: '4', imgUrl: './apricot.jpg' },
  { text: 'Grape', id: '5', imgUrl: './grape.jpg' },
  { text: 'Strawberry', id: '6', imgUrl: './strawberry.jpg' },
  { text: 'Pineapple', id: '7', imgUrl: './pineapple.jpg' },
  { text: 'Melon', id: '8', imgUrl: './melon.jpg' },
  { text: 'Lemon', id: '9', imgUrl: './lemon.jpg' },
  { text: 'Cherry', id: '10', imgUrl: './cherry.jpg' },
];
//Initialize ListView control
let listViewInstance: ListView = new ListView({
  //set the data to datasource property
  dataSource: fruitsdata.slice(),
  //set the template for list items
  template: '<div class="fruits"><div class="first"><button class="delete e-control e-btn e-small
```

```

e-round e-delete-btn e-primary e-icon-btn" data-ripple="true"><span
class="e-btn-icon e-icons delete-icon"></span></button></div><div
class="fruitName">${text}</div></div>',
    //set sortOrder for list items
    sortOrder: 'Ascending',
    actionComplete: () => {
        wireEvents();
    }
});
//Render initialized ListView
listViewInstance.appendTo('#element');
let dialogObj: Dialog = new Dialog({
    header: 'Add fruit',
    content: '<div id="listDialog"><div class="input_name"><label
for="name">Fruit Name: </label><input id="name" class="e-input" type="text"
placeholder="Enter fruit name"/></div><div><label for="imgurl">Fruit Image:
</label><input id="imgurl" class="e-input" type="text" placeholder="Enter
image url"/></div></div>',
    showCloseIcon: true,
    buttons: [{
        click: dlgButtonClick,
        buttonModel: { content: 'Add', isPrimary: true }
    }],
    width: '300px',
    visible: false
});
dialogObj.appendTo('#dialog');
function addItem() {
    (document.getElementById("name") as HTMLInputElement).value = "";
    (document.getElementById("imgurl") as HTMLInputElement).value = "";
    dialogObj.show()
}
function wireEvents() {
    Array.prototype.forEach.call(document.getElementsByClassName('e-delete-
btn'), (ele: HTMLButtonElement) => {
        ele.addEventListener('click', onDeleteBtnClick);
    });
    document.getElementById("add").addEventListener('click', addItem);
    document.getElementById("sort").addEventListener('click', sortItems);
    document.getElementById("search").addEventListener("keyup", onKeyUp);
}
//Here we are removing list item
function onDeleteBtnClick(e: MouseEventArgs) {
    e.stopPropagation();
    let li: Element = closest(e.currentTarget as Element, '.e-list-item');
    let data = listViewInstance.findItem(li);
    listViewInstance.removeItem(data as any);
    new DataManager(fruitsdata).remove('id', { id: (<{ [key: string]: any
}>data) ["id"] });
}
//Here we are adding list item
function dlgButtonClick() {
    let name: string = (document.getElementById("name") as
HTMLInputElement).value;
    let url: string = (document.getElementById("imgurl") as
HTMLInputElement).value;
    let id: number = Math.random() * 10000;

```

```

        listViewInstance.addItem([{ text: name, id: id, imgUrl: url }]);
        fruitsdata.push({ text: name, id: id, imgUrl: url });
        listViewInstance.element.querySelector('[data-uid="' + id +
        '"]').getElementsByClassName('e-delete-btn')[0].addEventListener('click',
        onDeleteBtnClick);
        dialogObj.hide();
    }
    //Here we are sorting list item
    function sortItems() {
        let ele: Element = document.getElementById("sort").firstElementChild;
        let des = ele.classList.contains(desClass) ? true : false;
        if (des) {
            ele.classList.remove(desClass);
            ele.classList.add(ascClass);
            listViewInstance.sortOrder = 'Ascending'
        } else {
            ele.classList.remove(ascClass);
            ele.classList.add(desClass);
            listViewInstance.sortOrder = 'Descending'
        }
        listViewInstance.dataBind();
        wireEvents();
    }
    //Here, the list items are filtered using the DataManager instance.
    function onKeyUp() {
        let value: string = (document.getElementById("search") as
        HTMLInputElement).value;
        let data: Object[] = new DataManager(fruitsdata).executeLocal(
            new Query().where("text", "startswith", value, true)
        );
        if (!value) {
            listViewInstance.dataSource = fruitsdata.slice();
        } else {
            listViewInstance.dataSource = data as { [key: string]: Object }[];
            listViewInstance.dataBind();
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
    Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="headerContainer">
            <div class="e-input-group">
                <input id="search" class="e-input" type="text"
placeholder="Search fruits">
                <span class="e-input-group-icon e-input-search"></span>
            </div>
            <button id="sort" class="e-control e-btn e-small e-round e-
primary e-icon-btn" title="Sort fruits" data-ripple="true">
                <span class="e-btn-icon e-icons e-sort-icon-
ascending"></span>
            </button>
            <button id="add" class="e-control e-btn e-small e-round e-
primary e-icon-btn" title="Add fruit" data-ripple="true">
                <span class="e-btn-icon e-icons e-add-icon"></span>
            </button>
            <div id="dialog"></div>
        </div>
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize listview with dynamic tags in EJ2 JavaScript Listview control

You can customize the ListView items using the [template](#) property. Here, the dynamic tags are added and removed in the list item from another ListView. Refer to the following steps to achieve this.

- Render the ListView with data source, and add button element with each list item of ListView on [actionComplete](#) event. Refer to the following code sample of actionComplete event.

```
`ts
```

```
// The actionComplete event for first ListView to add the button
```

```
function addButton() {
```

```
let buttonObj: { [key: string]: Object } = { obj: Button, prop: { iconCss: 'e-icons e-add-icon', cssClass: 'e-
small e-round' } };
```



```

let ele: HTMLCollection = document.getElementsByClassName("e-but");
for (let i: number = 0; i < ele.length; i++) {
  buttonObj.obj = new Button(buttonObj.prop);
  (buttonObj.obj as Button).appendTo(ele[i] as HTMLElement);
}
}
,

```

- Initialize dynamic ListView with required property that holds the tags of parent ListView, and bind the [select](#) event (triggers when the list item is selected), in which you can get and add the selected item value as tags into parent

ListView. Refer to the following code sample.

```

`ts
//Select the event that is is rendered inside dialog for ListView
function addTag(e: SelectEventArgs) {
  let listTag: HTMLSpanElement = document.createElement('span');
  listTag.className = 'advanced-option';
  let labelElem: HTMLSpanElement = document.createElement('span');
  labelElem.className = 'label';
  let deleteElem: HTMLSpanElement = document.createElement('span');
  deleteElem.className = 'delete';
  deleteElem.onclick = removeTag;
  labelElem.innerHTML = e.text;
  listTag.appendChild(labelElem);
  listTag.appendChild(deleteElem);
  let tag: HTMLSpanElement = document.createElement('span');
  tag.className = 'advanced-option-list';
  tag.appendChild(listTag);
  listViewInstance.element.querySelector('.e-active').appendChild(tag);
}
,

```

- Render the dialog control with empty content and append the created dynamic ListView object to the dialog on [created](#) event.

- Bind the click event for button icon (+) to update the ListView data source with tags, and open the dialog with this dynamic ListView. Refer to the following code sample.

```
`ts
//Method to hide/show the dialog and update the ListView data source
function renderDialog(id: string): void {
  if (document.getElementsByClassName('e-popup-open').length != 0) {
    dialog.hide();
  }
  else {
    listObj.dataSource = datasource[id];
    listObj.dataBind();
    dialog.show();
  }
}
```

- Bind the click event with added dynamic tags to remove it. Refer to the following code sample.

```
`ts
//Method to remove the list item
function removeTag() {
  this.parentNode.parentNode.remove();
}
```

INDEX.TS

```
import { ListView, SelectEventArgs } from '@syncfusion/ej2-lists';
import { Button } from '@syncfusion/ej2-buttons';
import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple, MouseEventArgs } from '@syncfusion/ej2-base';
enableRipple(false);
//Define customized template
let template: string = '<div><span class="templatetext">${Name} </span>
<span class="designationstyle"><button id ="${Id}" class="e-
but"></button></span></div>';
let data: { [key: string]: Object }[] = [{ "Id": "Brooke", "Name": "Brooke"
},
{ "Id": "Claire", "Name": "Claire" },
{ "Id": "Erik", "Name": "Erik" },
{ "Id": "Grace", "Name": "Grace" },
{ "Id": "Jacob", "Name": "Jacob" }];
let listViewInstance: ListView = new ListView({
```

```

//Set defined data to the dataSource property
dataSource: data,
//Set defined customized template
template: template,
//Set the fields property
fields: { text: 'Name' },
//Set the width to the ListView
width: 350,
//Bind the event to customize the ListView
actionComplete: addButton
});
//Render the initialized ListView control
listviewInstance.appendTo('#templatelist');
//actionComplete event for the first ListView
function addButton() {
    let buttonObj: { [key: string]: Object } = { obj: Button, prop: {
iconCss: 'e-icons e-add-icon', cssClass: 'e-small e-round' } };
    let ele: HTMLCollection = document.getElementsByClassName("e-but");
    for (let i: number = 0; i < ele.length; i++) {
        buttonObj.obj = new Button(buttonObj.prop);
        (buttonObj.obj as Button).appendTo(ele[i] as HTMLElement);
    }
}
//The click event for rendered button
for (let i: number = 0; i < data.length; i++) {
    document.getElementById(data[i].Id as string).addEventListener("click",
(e: MouseEventArgs) => {
        renderDialog((e.currentTarget as HTMLElement).id);
    });
}
let brookeTag: { [key: string]: Object }[] = [{ "id": "list11", "Name":
"Discover Music" },
{ "id": "list12", "Name": "Sales and Events" },
{ "id": "list13", "Name": "Categories" },
{ "id": "list14", "Name": "MP3 Albums" },
{ "id": "list15", "Name": "More in Music" },
];
let claireTag: { [key: string]: Object }[] = [{ "id": "list21", "Name":
"Songs" },
{ "id": "list22", "Name": "Bestselling Albums" },
{ "id": "list23", "Name": "New Releases" },
{ "id": "list24", "Name": "Bestselling Songs" },
];
let erikTag: { [key: string]: Object }[] = [{ "id": "list31", "Name":
"Artwork" },
{ "id": "list32", "Name": "Abstract" },
{ "id": "list33", "Name": "Acrylic Mediums" },
{ "id": "list34", "Name": "Creative Acrylic" },
{ "id": "list35", "Name": "Canvas Art" },
];
let graceTag: { [key: string]: Object }[] = [{ "id": "list41", "Name":
"Rock" },
{ "id": "list42", "Name": "Gospel" },
{ "id": "list43", "Name": "Latin Music" },
{ "id": "list44", "Name": "Jazz" },
];

```

```

let jacobTag: { [key: string]: Object }[] = [{ "id": "list51", "Name": "100
Albums - $5 Each" },
{ "id": "list52", "Name": "Hip-Hop and R&B Sale" },
{ "id": "list53", "Name": "CD Deals" }
];
let datasource: { [key: string]: { [key: string]: Object }[] } = { "Brooke":
brookeTag, "Claire": claireTag, "Erik": erikTag, "Grace": graceTag, "Jacob":
jacobTag };
//Initialize the ListView that is needed to append inside the dialog
let listObj: ListView = new ListView({
    showHeader: true,
    headerTitle: 'Favorite',
    width: '200px',
    dataSource: datasource.Brooke,
    fields: { text: 'Name' },
    select: addTag
});
//Initialize the dialog control
let dialog: Dialog = new Dialog({
    width: '200px',
    content: '<div id="list"></div>',
    animationSettings: { effect: 'None' },
    visible: false,
    created: createList,
    showCloseIcon: true,
    position: { X: document.querySelector('.e-add-
icon').getBoundingClientRect().left + 50, Y: document.querySelector('.e-add-
icon').getBoundingClientRect().top - 5 }
});
//Render the initialized dialog control
dialog.appendTo('#dialog');
//Method to hide/show the dialog and update the ListView data source
function renderDialog(id: string): void {
    if (document.getElementsByClassName('e-popup-open').length != 0) {
        dialog.hide();
    }
    else {
        listObj.dataSource = datasource[id];
        listObj.dataBind();
        dialog.show();
    }
}
//Created event for dialog
function createList() {
    let listElem: any =
document.getElementById('dialog').querySelector("#list");
    listObj.appendTo(listElem);
}
//Select event for ListView that is rendered inside the dialog
function addTag(e: SelectEventArgs) {
    let listTag: HTMLSpanElement = document.createElement('span');
    listTag.className = 'advanced-option';
    let labelElem: HTMLSpanElement = document.createElement('span');
    labelElem.className = 'label';
    let deleteElem: HTMLSpanElement = document.createElement('span');
    deleteElem.className = 'delete';
    deleteElem.onclick = removeTag;

```

```

    labelElem.innerHTML = e.text;
    listTag.appendChild(labelElem);
    listTag.appendChild(deleteElem);
    let tag: HTMLSpanElement = document.createElement('span');
    tag.className = 'advanced-option-list';
    tag.appendChild(listTag);
    listViewInstance.element.querySelector('.e-active').appendChild(tag);
}
//Method to remove the list item
function removeTag() {
    this.parentNode.parentNode.remove();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="sample">
            <!-- ListView element -->
            <div id="templatelist" tabindex="1"></div>
            <!-- Dialog element -->
            <div id="dialog"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Fetch selected items from listview template sample in EJ2 JavaScript Listview control

Single or multiple items can be selected by users in the ListView control.

By default, dataSource id and text is mapped in default rendering of listview, since it returns the selected item data properly. But in the custom template, dataSource and the corresponding mapping (text, id, elements rendered inside li element) will vary as per the application requirement.

So, we need to map id attribute to listview items using [fields](#) of [dataSource](#) to get the selected item data properly while working with custom templates. Refer to the below code snippet for template sample.

```
`ts
// Initialize ListView control
let listObj: ListView = new ListView({
//Set defined data to dataSource property
dataSource: dataSource,
//Map the appropriate columns to fields property
fields: { text: 'Name', id:'Name'},
//Set customized template
template: template,
enableRtl: true,
//ListView Select event
select: onSelect
});
//Render initialized ListView control
listObj.appendTo('#listview');
`
```

INDEX.TS

```
import { ListView } from '@syncfusion/ej2-lists';
let template: any = '<div class="settings">'
+ '<div id="postContainer"><div id="postImg">'
+ '<img src=${image} style="height:35px;width:35px;border-
radius:50%; border: 1px solid #ccc;" /></div>'
+ '<div id="content">'
+ '<div class="name">${Name}</div>'
+ '<div id="info">${contact}</div>'
+ '</div>'
+ '</div>'
//Define an array of JSON data
let dataSource: any = [
{ Name: 'Nancy', contact: '(206) 555-985774', id: '1', image:
'https://ej2.syncfusion.com/demos/src/grid/images/1.png', category:
'Experience' },
```

```

    { Name: 'Janet', contact: '(206) 555-3412', id: '2', image:
'https://ej2.syncfusion.com/demos/src/grid/images/3.png', category:
'Fresher' },
    { Name: 'Margaret', contact: '(206) 555-8122', id: '4', image:
'https://ej2.syncfusion.com/demos/src/grid/images/4.png', category:
'Experience' },
    { Name: 'Andrew ', contact: '(206) 555-9482', id: '5', image:
'https://ej2.syncfusion.com/demos/src/grid/images/2.png', category:
'Experience' },
    { Name: 'Steven', contact: '(71) 555-4848', id: '6', image:
'https://ej2.syncfusion.com/demos/src/grid/images/5.png', category:
'Fresher' },
  ];
  // Initialize ListView control
  let listObj: ListView = new ListView({
    //Set defined data to dataSource property
    dataSource: dataSource,
    //Map the appropriate columns to fields property
    fields: { text: 'Name', id: 'Name' },
    //Set customized template
    template: template,
    enableRtl: true,
    select: onSelect
  });
  //Render initialized ListView control
  listObj.appendTo('#listview');
  function onSelect() {
    let selectedItem = listObj.getSelectedItems();
    document.getElementById('val').innerHTML = 'Selected Item :
<b>' + selectedItem.text + '</b>';
  }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for ListView </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for ListView UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <!-- ListView element -->
  <div id="listview" tabindex="1">
  </div>
  <table id="val"></table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Trace events of listview in EJ2 JavaScript Listview control

The ListView control triggers events based on its actions. The events can be used as extension points to perform

custom operations. Refer to the following steps to trace the ListView events:

1. Render the ListView with [dataSource](#), and bind the [actionBegin](#), [actionComplete](#), and [select](#) events.
2. Perform custom operations in actionBegin, actionComplete, and select events.
3. Provide event log details for actionBegin and actionComplete events, and they will be displayed in the event trace panel when the ListView action starts and the dataSource bound successfully.
4. Get the selected item details from the [SelectEventArgs](#) in the select event, and display the selected list item text in the event trace panel while selecting list items.
5. Use clear button to remove event trace information.

INDEX.TS

```

import { ListView, SelectEventArgs } from "@syncfusion/ej2-lists";
import { Button } from "@syncfusion/ej2-buttons";
//Define an array of JSON data
let data: { [key: string]: Object }[] = [
  { text: "Hennessey Venom", id: "list-01" },
  { text: "Bugatti Chiron", id: "list-02" },
  { text: "Bugatti Veyron Super Sport", id: "list-03" },
  { text: "SSC Ultimate Aero", id: "list-04" },
  { text: "Koenigsegg CCR", id: "list-05" },
  { text: "McLaren F1", id: "list-06" },
  { text: "Aston Martin One- 77", id: "list-07" },
  { text: "Jaguar XJ220", id: "list-08" },
  { text: "McLaren P1", id: "list-09" },
  { text: "Ferrari LaFerrari", id: "list-10" }
];
let clear: Button = new Button();
clear.appendTo('#clear');
//Initialize the ListView control
let listObj: ListView = new ListView({

```



```

//Set the defined data to the dataSource property
dataSource: data,
actionBegin: onActionBegin,
actionComplete: onActionComplete,
select: onSelect,
width:"250"
});
//Render the initialized ListView control
listObj.appendTo("#listview-def");
//Clears the event log details
document.getElementById("clear").onclick = () => {
    document.getElementById("EventLog").innerHTML = "";
};
//Handler for actionBegin event trace
function onActionBegin(): void {
    appendElement("<b>actionBegin </b> event is triggered<hr>");
}
//Handler for select event trace
function onSelect(args: SelectEventArgs): void {
    appendElement(args.text + "<b>##160;##160;is selected</b><hr>");
}
//Handler for actionComplete event trace
function onActionComplete(): void {
    appendElement("<b>actionComplete</b> is triggered <hr>");
}
//Display event log
function appendElement(html: string): void {
    let span: HTMLElement = document.createElement("span");
    span.innerHTML = html;
    let log: HTMLElement = document.getElementById("EventLog");
    log.insertBefore(span, log.firstChild);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

  <div id="container">
    <h4 id="evt-text">
      <b>Event Trace</b>
    </h4>
    <div id="list-container">
      <!-- ListView element -->
      <div id="listview-def" tabindex="1">
        </div>
        <div id="list_event">
          <div id="evt">
            <div class="eventarea" style="height:273px;overflow:
auto">
              <!-- Event log element -->
              <span class="EventLog" id="EventLog" style="word-
break: normal;"></span>
            </div>
            <div class="evtbtn">
              <!-- clear button element -->
              <input id="clear" type="button" value="Clear">
            </div>
          </div>
        </div>
      </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
  </body></html>

```

Customize nested listview as bread crumbs in EJ2 JavaScript Listview control

ListView header can be customized using [headerTemplate](#) property. Here We customized the header of nested ListView as BreadCrumbs with headerTemplate property . i.e while navigating to child data of a list item, the header data is customized with the selected data path. For example, the header of nested ListView is Continent. While selecting a list item(Asia) then the header will be customized as Continent>Asia.

- This customization achieved while front and back navigations of list items with `actionComplete` event of ListView.
- On actionComplete we append the selected text in header element.
- And in back navigation, we removed the last appended span from header template

And also we can able to navigate the desired child level by clicking list items appended in the customized header. For example, let consider header of nested ListView is `Continent>Asia>India`. If we want to navigate to Continent level of ListView, then we can click the Continent in Header

Note: In each navigation we have calculated the appended span tag length in `calculateLevelForElements` method to update header.

INDEX.TS

```

import { ListView } from '@syncfusion/ej2-lists';
//define the array of JSON
let continent: { [key: string]: Object }[] = [
    {
        'text': 'Asia',
        'id': '01',
        'category': 'Continent',
        'child': [{
            'text': 'India',
            'id': '1',
            'category': 'Asia',
            'child': [{
                'text': 'Delhi',
                'id': '1001',
                'category': 'India',
            },
            {
                'text': 'Kashmir',
                'id': '1002',
                'category': 'India',
            },
            {
                'text': 'Goa',
                'id': '1003',
                'category': 'India',
            },
        ]
    },
    {
        'text': 'China',
        'id': '2',
        'category': 'Asia',
        'child': [{
            'text': 'Zhejiang',
            'id': '2001',
            'category': 'China',
        },
        {
            'text': 'Hunan',
            'id': '2002',
            'category': 'China',
        },
        {
            'text': 'Shandong',
            'id': '2003',
            'category': 'China',
        }
    ]
    }
],
    {
        'text': 'North America',
        'id': '02',
        'category': 'Continent',
        'child': [{
            'text': 'USA',

```

```

        'id': '3',
        'category': 'North America',
        'child': [{
            'text': 'California',
            'id': '3001',
            'category': 'USA',
        },
        {
            'text': 'New York',
            'id': '3002',
            'category': 'USA',
        },
        {
            'text': 'Florida',
            'id': '3003',
            'category': 'USA',
        }
    ]
},
{
    'text': 'Canada',
    'id': '4',
    'category': 'North America',
    'child': [{
        'text': 'Ontario',
        'id': '4001',
        'category': 'Canada',
    },
    {
        'text': 'Alberta',
        'id': '4002',
        'category': 'Canada',
    },
    {
        'text': 'Manitoba',
        'id': '4003',
        'category': 'Canada',
    }
    ]
}]
},
{
    'text': 'Europe',
    'id': '03',
    'category': 'Continent',
    'child': [{
        'text': 'Germany',
        'id': '5',
        'category': 'Europe',
        'child': [{
            'text': 'Berlin',
            'id': '5001',
            'category': 'Germany',
        },
        {
            'text': 'Bavaria',
            'id': '5002',
            'category': 'Germany',
        },
    ]
    },
    {
        'text': 'France',
        'id': '6',
        'category': 'Europe',
        'child': [{
            'text': 'Paris',
            'id': '6001',
            'category': 'France',
        },
            {
                'text': 'Lyon',
                'id': '6002',
                'category': 'France',
            }
        ]
    },
    {
        'text': 'Italy',
        'id': '7',
        'category': 'Europe',
        'child': [{
            'text': 'Rome',
            'id': '7001',
            'category': 'Italy',
        },
            {
                'text': 'Venice',
                'id': '7002',
                'category': 'Italy',
            }
        ]
    },
    {
        'text': 'Spain',
        'id': '8',
        'category': 'Europe',
        'child': [{
            'text': 'Madrid',
            'id': '8001',
            'category': 'Spain',
        },
            {
                'text': 'Barcelona',
                'id': '8002',
                'category': 'Spain',
            }
        ]
    },
    {
        'text': 'United Kingdom',
        'id': '9',
        'category': 'Europe',
        'child': [{
            'text': 'London',
            'id': '9001',
            'category': 'United Kingdom',
        },
            {
                'text': 'Edinburgh',
                'id': '9002',
                'category': 'United Kingdom',
            }
        ]
    },
    {
        'text': 'Netherlands',
        'id': '10',
        'category': 'Europe',
        'child': [{
            'text': 'Amsterdam',
            'id': '10001',
            'category': 'Netherlands',
        },
            {
                'text': 'Rotterdam',
                'id': '10002',
                'category': 'Netherlands',
            }
        ]
    },
    {
        'text': 'Sweden',
        'id': '11',
        'category': 'Europe',
        'child': [{
            'text': 'Stockholm',
            'id': '11001',
            'category': 'Sweden',
        },
            {
                'text': 'Gothenburg',
                'id': '11002',
                'category': 'Sweden',
            }
        ]
    },
    {
        'text': 'Poland',
        'id': '12',
        'category': 'Europe',
        'child': [{
            'text': 'Warsaw',
            'id': '12001',
            'category': 'Poland',
        },
            {
                'text': 'Krakow',
                'id': '12002',
                'category': 'Poland',
            }
        ]
    },
    {
        'text': 'Czech Republic',
        'id': '13',
        'category': 'Europe',
        'child': [{
            'text': 'Prague',
            'id': '13001',
            'category': 'Czech Republic',
        },
            {
                'text': 'Brno',
                'id': '13002',
                'category': 'Czech Republic',
            }
        ]
    },
    {
        'text': 'Austria',
        'id': '14',
        'category': 'Europe',
        'child': [{
            'text': 'Vienna',
            'id': '14001',
            'category': 'Austria',
        },
            {
                'text': 'Salzburg',
                'id': '14002',
                'category': 'Austria',
            }
        ]
    },
    {
        'text': 'Switzerland',
        'id': '15',
        'category': 'Europe',
        'child': [{
            'text': 'Bern',
            'id': '15001',
            'category': 'Switzerland',
        },
            {
                'text': 'Zurich',
                'id': '15002',
                'category': 'Switzerland',
            }
        ]
    },
    {
        'text': 'Belgium',
        'id': '16',
        'category': 'Europe',
        'child': [{
            'text': 'Brussels',
            'id': '16001',
            'category': 'Belgium',
        },
            {
                'text': 'Antwerp',
                'id': '16002',
                'category': 'Belgium',
            }
        ]
    },
    {
        'text': 'Luxembourg',
        'id': '17',
        'category': 'Europe',
        'child': [{
            'text': 'Luxembourg',
            'id': '17001',
            'category': 'Luxembourg',
        },
            {
                'text': 'Esch-sur-Alzette',
                'id': '17002',
                'category': 'Luxembourg',
            }
        ]
    },
    {
        'text': 'Portugal',
        'id': '18',
        'category': 'Europe',
        'child': [{
            'text': 'Lisbon',
            'id': '18001',
            'category': 'Portugal',
        },
            {
                'text': 'Oporto',
                'id': '18002',
                'category': 'Portugal',
            }
        ]
    },
    {
        'text': 'Greece',
        'id': '19',
        'category': 'Europe',
        'child': [{
            'text': 'Athens',
            'id': '19001',
            'category': 'Greece',
        },
            {
                'text': 'Thessaloniki',
                'id': '19002',
                'category': 'Greece',
            }
        ]
    },
    {
        'text': 'Turkey',
        'id': '20',
        'category': 'Europe',
        'child': [{
            'text': 'Istanbul',
            'id': '20001',
            'category': 'Turkey',
        },
            {
                'text': 'Ankara',
                'id': '20002',
                'category': 'Turkey',
            }
        ]
    },
    {
        'text': 'Russia',
        'id': '21',
        'category': 'Europe',
        'child': [{
            'text': 'Moscow',
            'id': '21001',
            'category': 'Russia',
        },
            {
                'text': 'Saint Petersburg',
                'id': '21002',
                'category': 'Russia',
            }
        ]
    },
    {
        'text': 'Ukraine',
        'id': '22',
        'category': 'Europe',
        'child': [{
            'text': 'Kyiv',
            'id': '22001',
            'category': 'Ukraine',
        },
            {
                'text': 'Lviv',
                'id': '22002',
                'category': 'Ukraine',
            }
        ]
    },
    {
        'text': 'Belarus',
        'id': '23',
        'category': 'Europe',
        'child': [{
            'text': 'Minsk',
            'id': '23001',
            'category': 'Belarus',
        },
            {
                'text': 'Gomel',
                'id': '23002',
                'category': 'Belarus',
            }
        ]
    },
    {
        'text': 'Moldova',
        'id': '24',
        'category': 'Europe',
        'child': [{
            'text': 'Chisinau',
            'id': '24001',
            'category': 'Moldova',
        },
            {
                'text': 'Iasi',
                'id': '24002',
                'category': 'Moldova',
            }
        ]
    },
    {
        'text': 'Romania',
        'id': '25',
        'category': 'Europe',
        'child': [{
            'text': 'Bucharest',
            'id': '25001',
            'category': 'Romania',
        },
            {
                'text': 'Cluj-Napoca',
                'id': '25002',
                'category': 'Romania',
            }
        ]
    },
    {
        'text': 'Bulgaria',
        'id': '26',
        'category': 'Europe',
        'child': [{
            'text': 'Sofia',
            'id': '26001',
            'category': 'Bulgaria',
        },
            {
                'text': 'Plovdiv',
                'id': '26002',
                'category': 'Bulgaria',
            }
        ]
    },
    {
        'text': 'Serbia',
        'id': '27',
        'category': 'Europe',
        'child': [{
            'text': 'Belgrade',
            'id': '27001',
            'category': 'Serbia',
        },
            {
                'text': 'Novi Sad',
                'id': '27002',
                'category': 'Serbia',
            }
        ]
    },
    {
        'text': 'Croatia',
        'id': '28',
        'category': 'Europe',
        'child': [{
            'text': 'Zagreb',
            'id': '28001',
            'category': 'Croatia',
        },
            {
                'text': 'Split',
                'id': '28002',
                'category': 'Croatia',
            }
        ]
    },
    {
        'text': 'Slovenia',
        'id': '29',
        'category': 'Europe',
        'child': [{
            'text': 'Ljubljana',
            'id': '29001',
            'category': 'Slovenia',
        },
            {
                'text': 'Maribor',
                'id': '29002',
                'category': 'Slovenia',
            }
        ]
    },
    {
        'text': 'Hungary',
        'id': '30',
        'category': 'Europe',
        'child': [{
            'text': 'Budapest',
            'id': '30001',
            'category': 'Hungary',
        },
            {
                'text': 'Debrecen',
                'id': '30002',
                'category': 'Hungary',
            }
        ]
    },
    {
        'text': 'Slovakia',
        'id': '31',
        'category': 'Europe',
        'child': [{
            'text': 'Bratislava',
            'id': '31001',
            'category': 'Slovakia',
        },
            {
                'text': 'Kosice',
                'id': '31002',
                'category': 'Slovakia',
            }
        ]
    },
    {
        'text': 'Lithuania',
        'id': '32',
        'category': 'Europe',
        'child': [{
            'text': 'Vilnius',
            'id': '32001',
            'category': 'Lithuania',
        },
            {
                'text': 'Kaunas',
                'id': '32002',
                'category': 'Lithuania',
            }
        ]
    },
    {
        'text': 'Latvia',
        'id': '33',
        'category': 'Europe',
        'child': [{
            'text': 'Riga',
            'id': '33001',
            'category': 'Latvia',
        },
            {
                'text': 'Daugavpils',
                'id': '33002',
                'category': 'Latvia',
            }
        ]
    },
    {
        'text': 'Estonia',
        'id': '34',
        'category': 'Europe',
        'child': [{
            'text': 'Tallinn',
            'id': '34001',
            'category': 'Estonia',
        },
            {
                'text': 'Tartu',
                'id': '34002',
                'category': 'Estonia',
            }
        ]
    },
    {
        'text': 'Finland',
        'id': '35',
        'category': 'Europe',
        'child': [{
            'text': 'Helsinki',
            'id': '35001',
            'category': 'Finland',
        },
            {
                'text': 'Tampere',
                'id': '35002',
                'category': 'Finland',
            }
        ]
    },
    {
        'text': 'Iceland',
        'id': '36',
        'category': 'Europe',
        'child': [{
            'text': 'Reykjavik',
            'id': '36001',
            'category': 'Iceland',
        },
            {
                'text': 'Akureyri',
                'id': '36002',
                'category': 'Iceland',
            }
        ]
    },
    {
        'text': 'Norway',
        'id': '37',
        'category': 'Europe',
        'child': [{
            'text': 'Oslo',
            'id': '37001',
            'category': 'Norway',
        },
            {
                'text': 'Bergen',
                'id': '37002',
                'category': 'Norway',
            }
        ]
    },
    {
        'text': 'Denmark',
        'id': '38',
        'category': 'Europe',
        'child': [{
            'text': 'Copenhagen',
            'id': '38001',
            'category': 'Denmark',
        },
            {
                'text': 'Aarhus',
                'id': '38002',
                'category': 'Denmark',
            }
        ]
    },
    {
        'text': 'Sweden',
        'id': '39',
        'category': 'Europe',
        'child': [{

```

```

        {
            'text': 'Hesse',
            'id': '5003',
            'category': 'Germany',
        }
    ], {
        'text': 'France',
        'id': '6',
        'category': 'Europe',
        'child': [{
            'text': 'Paris',
            'id': '6001',
            'category': 'France',
        },
        {
            'text': 'Lyon',
            'id': '6002',
            'category': 'France',
        },
        {
            'text': 'Marseille',
            'id': '6003',
            'category': 'France',
        }
    ]
    }],
    {
        'text': 'Australia',
        'id': '04',
        'category': 'Continent',
        'child': [{
            'text': 'Australia',
            'id': '7',
            'category': 'Australia',
            'child': [{
                'text': 'Sydney',
                'id': '7001',
                'category': 'Australia',
            },
            {
                'text': 'Melbourne',
                'id': '7002',
                'category': 'Australia',
            },
            {
                'text': 'Brisbane',
                'id': '7003',
                'category': 'Australia',
            }
        ]
        }, {
            'text': 'New Zealand',
            'id': '8',
            'category': 'Australia',
            'child': [{
                'text': 'Milford Sound',
                'id': '8001',
                'category': 'New Zealand',
            }
        ]
    }

```

```

        },
        {
            'text': 'Tongariro National Park',
            'id': '8002',
            'category': 'New Zealand',
        },
        {
            'text': 'Fiordland National Park',
            'id': '8003',
            'category': 'New Zealand',
        }
    ]
}
},
{
    'text': 'Africa',
    'id': '05',
    'category': 'Continent',
    'child': [{
        'text': 'Morocco',
        'id': '9',
        'category': 'Africa',
        'child': [{
            'text': 'Rabat',
            'id': '9001',
            'category': 'Morocco',
        },
        {
            'text': 'Toubkal',
            'id': '9002',
            'category': 'Morocco',
        },
        {
            'text': 'Todgha Gorge',
            'id': '9003',
            'category': 'Morocco',
        }
    ]
}], {
    'text': 'South Africa',
    'id': '10',
    'category': 'Africa',
    'child': [{
        'text': 'Cape Town',
        'id': '10001',
        'category': 'South Africa',
    },
    {
        'text': 'Pretoria',
        'id': '10002',
        'category': 'South Africa',
    },
    {
        'text': 'Bloemfontein',
        'id': '10003',
        'category': 'South Africa',
    }
    ]
}
}

```

```

];
let text:string, backBtn:HTMLElement, title, headerElement, hasChild =
false;
//Initialize ListView component
let listViewInstance = new ListView({
    //set the data to dataSource property
    dataSource: continent,
    // map the groupBy field with category column
    fields: { tooltip: "text" },
    headerTemplate: '<div class="header-
content"><span>Continent</span></div>',
    showHeader: true,
    select: onSelect,
    actionComplete: onComplete
});
//Render initialized ListView
listviewInstance.appendTo("#listview");

backBtn = document.querySelector('.e-back-button') as HTMLElement;
title = document.querySelector(".header-content");
title.addEventListener("click", navigateBack);

function onSelect(e: any) {
    text = e.text;
    hasChild = !e.item.classList.contains("e-has-child");
}

function onComplete() {
    if (!hasChild) {
        headerElement = document.querySelector(".header-content").innerHTML;
        title = document.querySelector(".nested-header .header-content");
        if (headerElement && text != undefined) {
            // Create element with new clicked item in header
            let element = document.createElement("span");
            element.textContent = ` > ${text}`;
            title.appendChild(element);

            // Recalculate levels, Since an element is added
            calculateLevelForElements();
        }
    }
}

backBtn.addEventListener("click", function () {
    let elements = document.querySelectorAll(
        ".nested-header .header-content span"
    );
    elements[elements.length - 1].remove();
    // Recalculate levels, Since an element is removed
    calculateLevelForElements();
    if (backBtn.style.display == "none") {
        hasChild = false;
    } else {
        hasChild = true;
    }
});
// Calculate level for each header element
function calculateLevelForElements() {

```

```

let elements = document.querySelectorAll(
    ".nested-header .header-content span"
);
let index = 0;
for (let i = elements.length - 1; i >= 0; i--) {
    (elements[index] as any).level = i;
    index++;
}

// Navigate back event handler based on element's level
function navigateBack(event: any) {
    if (event.target && event.target.level) {
        for (let i = 0; i <= event.target.level; i++) {
            backBtn.click();
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="listview"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```


Integrate pager component with listview in EJ2 JavaScript Listview control

The first and foremost step is to obtain the **Pager** component from **Grid**. Install the ej2-grid package using the following command.

,

```
npm install @syncfusion/ej2-grids --save
```

,

Import the Pager to the ListView sample which has been created.

,

```
import { Pager } from "@syncfusion/ej2-grids";
```

,

The [totalRecordsCount](#) property of Pager must be specified whenever using this particular component. By using [pageSize](#) property, the number of list items to be displayed is made available. The [pageCount](#) property allows the user to specify the visibility of the page numbers accordingly. Since the paging sample in the upcoming code snippet uses these three properties, the explanation provided here were minimal and to the point. For further API concerns in Pager component, [click here](#).

With the help of the **query** property of ListView, the user can specify the number of records to be displayed in the current page.

The [query](#) property helps in splitting the entire datasource based on our convenience. In the sample provided below, when clicking the next button in pager, it fetches the datasource based on page size and current page of Pager component.

```
`ts
```

```
click: function () {
```

```
listObj.query= new Query().range((pager.currentPage - 1) * pager.pageSize, (pager.currentPage * pager.pageSize));
```

```
}
```

,

Note: When **pageSize** isn't mentioned, it defaults to 12 records per page.

INDEX.TS

```
import { ListView } from '@syncfusion/ej2-lists';
import { Pager } from '@syncfusion/ej2-grids';
import { DataManager, Query, JsonAdaptor } from '@syncfusion/ej2-data';
import { data } from './datasource.ts';
let pager: Pager = new Pager({
  totalRecordsCount: data.length,
  pageSize: 10,
  pageCount: 2,
  click: function () {
    listObj.query= new Query().range((pager.currentPage - 1) *
pager.pageSize, (pager.currentPage * pager.pageSize));
  }
});
```

```

pager.appendTo('#Pager');
let header: string = '<table class="w-100"> <tr><td class="w-25">Order
ID</td><td class="w-45">Ship Name</td><td class="w-
25">ShipCity</td></tr></table>';
let template: string = '<table class="w-100"> <colgroup><col
span="2"><col></colgroup><tr><td class="w-25">${OrderID}</td><td class="w-
45">${ShipName}</td><td class="w-25">${ShipCity}</td></tr></table>';
//Initialize ListView component
// Initialize the ListView control
let listObj: ListView = new ListView({
    //Set the defined data to the data source property
    dataSource: new DataManager({
        json: data,
        adaptor: new JsonAdaptor
    }),
    query: new Query().range(0, pager.pageSize),
    template: template,
    headerTemplate: header,
    showHeader: true
});
//Render the initialized ListView control
listObj.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for ListView </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for ListView UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

    <script src="./es5-datasource.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <div id="Pager"></div>

```

```

    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Maps

Populate data in EJ2 JavaScript Maps control

This section explains how to populate data inputs and provide it to the Maps component.

Geometry types

GeoJSON data contains geometry objects with properties such as geometry types and coordinates. The geometry types are the values present in the geometry objects of the GeoJSON data that specify the type of shape to be rendered, as well as the coordinates that help to draw the shape's boundary line. The supportive geometry types are:

| Shapes | Supported |

| --- | --- |

| Polygon | Yes |

| MultiPolygon | Yes |

| LineString | Yes |

| MultiLineString | Yes |

| Point | Yes |

| MultiPoint | Yes |

| GeometryCollection | Yes |

Shape data

The shape data collection describes geographical shape information that can be obtained from [GEOJSON format shapes](#). The Map shapes are rendered with this data. The custom shapes such as seat selection in bus, seat selection in a cricket stadium and more useful information can be also added to the geographical data.

Data source

The [dataSource](#) property is used to represent statistical data in the Maps component, and it accepts a collection of values as input. For example, a list of objects as input can be provided to the data source. This data source will be used to color the map, display data labels, and display tooltip, among other things.

The data source is populated with JSON data relative to shape data and stored in JSON object. The USA population as data source is used for better understanding. The **populationData.ts** file is used to store JSON data in JSON object **populationData**.

[populationData.ts]

```
`ts
export let populationData: object[] = [
{
'code': 'AF',
'value': 53,
'name': 'Afghanistan',
'population': 29863010,
'density': 119
},
{
'code': 'AL',
'value': 117,
'name': 'Albania',
'population': 3195000,
'density': 111
},
{
'code': 'DZ',
'value': 15,
'name': 'Algeria',
'population': 34895000,
'density': 15
},
{
'code': 'AO',
'value': 15,
'name': 'Angola',
'population': 18498000,
'density': 15
},
{
'code': 'AR',
'value': 15,
```

```
'name': 'Argentina',
'population': 40091359,
'density': 14
},
{
'code': 'AM',
'value': 109,
'name': 'Armenia',
'population': 3230100,
'density': 108
}
];
`
```

Data binding

The following properties in the [layers](#) are used for binding data in the Maps component. Both the properties are related to each other.

- `shapePropertyPath`
- `shapeDataPath`

Shape property path

The [shapePropertyPath](#) property is used to refer to the column name in the [shapeData](#) property of shape layers to identify the shape. When the values of [shapeDataPath](#) property from the [dataSource](#) property and [shapePropertyPath](#) property from the [shapeData](#) property match, then the associated object from the data source is bound to the corresponding shape.

`world-map.js` file contains following data and its field **name** value is used to map the corresponding shape with the provided data source.

```
`javascript
export var world_map = {
"type": "Feature",
"properties": {
"admin": "Afghanistan",
"name": "Afghanistan",
"continent": "Asia"
},
"geometry": { "type": "Polygon", "coordinates": [[[61.21081709172573, ... ],
...

```

Shape data path

The [shapeDataPath](#) property is similar to the [shapePropertyPath](#) property, but it refers to the field name in the [dataSource](#) property. For example, [populationData](#) contains the **code**, **value**, **name**, **population** and **density** fields. Here, the **name** field is set to the shapeDataPath to map the corresponding name field value of shape data.

In the below example, both **name** fields contain the same value as **Afghanistan**, this value is matched in both shape data and data source, so that the details associated with **Afghanistan** will be mapped to the corresponding shape and used to color the corresponding shape, display data labels, display tooltips, and more.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    dataSource: [
      {
        'code': 'AF',
        'value': 53,
        'name': 'Afghanistan',
        'population': 29863010,
        'density': 119,
        'color': '#DEEBAE'
      },
      {
        'code': 'AL',
        'value': 117,
        'name': 'Albania',
        'population': 3195000,
        'density': 111,
        'color': '#A4D6AD'
      },
      {
        'code': 'DZ',
        'value': 15,
        'name': 'Algeria',
        'population': 34895000,
        'density': 15,
        'color': '#37AFAB'
      },
      {
        'code': 'AO',
        'value': 15,
        'name': 'Angola',
        'population': 18498000,
        'density': 15,
        'color': '#547C84'
      },
      {
        'code': 'AR',
        'value': 15,
```

```

        'name': 'Argentina',
        'population': 40091359,
        'density': 14,
        'color': '#CEBF93'
    },
    {
        'code': 'AM',
        'value': 109,
        'name': 'Armenia',
        'population': 3230100,
        'density': 108,
        'color': '#a69d70'
    }
],
shapeSettings: {
    colorValuePath: 'color',
    fill: '#E5E5E5'
}
}]
});
map.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Binding complex data source

To bind the data field from data source to the maps in two different ways.

1. Bind the field name directly to the properties as [shapeDataPath](#), [colorValuePath](#), [valuePath](#) and [shapeValuePath](#).
2. Bind the field name as `data.field` to the properties as [shapeDataPath](#), [colorValuePath](#), [valuePath](#) and [shapeValuePath](#).

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [
    {
      shapeData: world_map,
      shapeDataPath: 'data.continent',
      shapePropertyPath: 'continent',
      dataSource: [
        { "Continent": "North America", 'color': '#71B081',
          data: { "continent": "North America", 'color': '#71B081' } },
        { "Continent": "South America", 'color': '#5A9A77',
          data: { "continent": "South America", 'color': '#5A9A77' } },
        { "Continent": "Africa", 'color': '#498770',
          data: { "continent": "Africa", 'color': '#498770' } },
        { "Continent": "Europe", 'color': '#39776C',
          data: { "continent": "Europe", 'color': '#39776C' } },
        { "Continent": "Asia", 'color': '#266665',
          data: { "continent": "Asia", 'color': '#266665' } },
        { "Continent": "Australia", 'color': '#124F5E',
          data: { "continent": "Australia", 'color': '#124F5E' } }
      ],
      shapeSettings: {
        colorValuePath: 'data.color',
      },
      tooltipSettings: {
        visible: true,
        valuePath: 'data.continent'
      },
      bubbleSettings: [
        {
          visible: true,
          valuePath: 'data.value',
          colorValuePath: 'data.color',
          animationDuration: 0,
          minRadius: 20,
          maxRadius: 90,
          opacity: 0.8,
          dataSource: [
            { 'name': 'India', 'value': 18.89685398845257,
              'population': 391292635,
```



```

        data: { 'color': 'red', 'population': 391292635,
'value': 189685398845257 }
    },
    tooltipSettings: {
        visible: true,
        valuePath: 'data.population',
        template:"<div>${data.population}</div>"
    }
},
markerSettings: [
    {
        visible: true,
        dataSource: [
            { latitude: 37.6276571, longitude: -122.4276688, name:
'San Bruno',
                data: { x: 37.6276571, y: -122.4276688, name: 'San
Bruno', shape: 'Pentagon',
                    color: 'red', imageUrl: 'images/ballon.png' }
            },
            { latitude: 33.5302186, longitude: -117.7418381, name:
'Laguna Niguel',
                data: { x: 33.5302186, y: -117.7418381, name: 'Laguna
Niguel', color: 'blue',
                    shape: 'Pentagon', imageUrl: 'images/ballon.png' }
            }
        ],
        shapeValuePath: "data.shape",
        colorValuePath: "data.color",
        height: 20,
        width: 20,
        offset: {
            y: -10,
            x: 0
        },
        longitudeValuePath: "data.y",
        latitudeValuePath: "data.x",
        tooltipSettings: {
            visible: true,
            valuePath: 'data.name',
            format: "${data.name}: ${data.x} : ${data.y}"
        },
        animationDuration: 0
    }
]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<script src="world-map.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Layers in EJ2 JavaScript Maps control

The Maps component is rendered through [layers](#) and any number of layers can be added to the Maps.

Multilayer

The Multilayer support allows loading multiple shapefiles and map providers in a single container, enabling Maps to display more information. The shape layer or map providers are the main layers of the Maps. Multiple layers can be added as **SubLayer** over the main layers using the [type](#) property in the [layers](#).

SubLayer

Sublayer is a type of shape file layer. It allows loading multiple shape files in a single map view. For example, a sublayer can be added over the main layer to view geographic features such as rivers, valleys and cities in a map of a country. Similar to the main layer, elements in the Maps such as markers, bubbles, color mapping and legends can be added to the sub-layer.

In this example, the United States map shape is used as shape data by utilizing the **usa.ts** file, and the **texas.ts** and **california.ts** files are used as sub-layers in the United States map.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [
    {
      shapeData: usa_map,
      shapeSettings: {
        fill: '#E5E5E5',
        border: {
          color: 'black',
          width: 0.1
        }
      }
    }
  ]
});

```

```

    },
    {
      shapeData: texas,
      type: 'SubLayer',
      shapeSettings: {
        fill: 'rgba(141, 206, 255, 0.6)',
        border: {
          color: '#1a9cff',
          width: 0.25
        }
      }
    },
    {
      shapeData: california,
      type: 'SubLayer',
      shapeSettings: {
        fill: 'rgba(141, 206, 255, 0.6)',
        border: {
          color: '#1a9cff',
          width: 0.25
        }
      }
    }
  ]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="usa.js"></script>
  <script src="california.js"></script>
  <script src="texas.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Displaying layer in the view

Multiple shape files and map providers can be loaded simultaneously in Maps. The [baseLayerIndex](#) property is used to determine which layer on the user interface should be displayed. This property is used for the Maps drill-down feature, so when the [baseLayerIndex](#) value is changed, the corresponding shape is loaded. In this example, two layers can be loaded with the World map and the United States map. Based on the given [baseLayerIndex](#) value the corresponding shape will be loaded in the user interface. If the [baseLayerIndex](#) value is set to **0**, then the world map will be loaded.

INDEX.JS

```

var map = new ej.maps.Maps({
  baseLayerIndex: 1,
  layers: [
    {
      shapeData: world_map,
    },
    {
      shapeData: usa_map
    }
  ]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body>
</html>
```

See Also

- [Adding multiple layers in map](#)
- [Custom path map](#)

Providers

Map provider in EJ2 JavaScript Maps control

The OpenStreetMap (OSM) is the online Maps provider built by a community of developers; it is free to use under an open license. It allows to view geographical data in a collaborative way from anywhere on the earth. The OSM Maps provides small tile images based on our requests and combines those images into a single image to display the Maps area in the Maps component.

Adding OpenStreetMap

The OSM Maps can be rendered using the [urlTemplate](#) property.

INDEX.JS

```
var maps = new ej.maps.Maps ({
  layers: [
    {
      urlTemplate: "https://tile.openstreetmap.org/level/tileX/tileY.png"
    }
  ],
});
maps.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="africa.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="element">
  </div>
  <script>
    var ele = document.getElementById('container');
```

```

        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Changing the tile server of the OpenStreetMap

The OSM tile server can be changed by setting the tile URL in the [urlTemplate](#) property. For more details about the OSM tile server, refer [here](#).

Enabling zooming and panning

The OSM Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a Maps for in-depth analysis. Panning helps to move a Maps around to focus the targeted area.

INDEX.JS

```

var maps = new ej.maps.Maps({
    zoomSettings: {
        enable: true,
        toolBars: ["Zoom", "ZoomIn", "ZoomOut", "Pan", "Reset"]
    },
    layers: [
        {
            urlTemplate: "https://tile.openstreetmap.org/level/tileX/tileY.png"
        }
    ]
});
maps.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="africa.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="element">
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {

```

```

        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Adding markers and navigation line

Markers can be added to the layers of OSM Maps by setting the corresponding location's coordinates of latitude and longitude using [markerSettings](#). Navigation lines can be added on top of an OSM Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [navigationLineSettings](#).

INDEX.JS

```

var maps = new ej.maps.Maps({
    zoomSettings: {
        zoomFactor: 4
    },
    centerPosition: {
        latitude: 29.394708,
        longitude: -94.954653
    },
    layers: [
        {
            urlTemplate: "https://tile.openstreetmap.org/level/tileX/tileY.png",
            markerSettings: [{
                visible: true,
                height: 25,
                width: 15,
                dataSource: [
                    {
                        latitude: 34.060620,
                        longitude: -118.330491,
                        name: "California"
                    },
                    {
                        latitude: 40.724546,
                        longitude: -73.850344,
                        name: "New York"
                    }
                ]
            }],
            navigationLineSettings: [{
                visible: true,
                color: "blue",
                width: 5,
                angle: 0.1,
                latitude: [34.060620, 40.724546],
                longitude: [-118.330491, -73.850344]
            }
        ]
    ]
});
maps.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="africa.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="element">
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the OSM Maps layer for highlighting a particular continent or country in OSM Maps by adding another layer and specifying the [type](#) property of Maps layer to **SubLayer**.

INDEX.JS

```

var maps = new ej.maps.Maps({
  layers: [
    {
      urlTemplate:"https://tile.openstreetmap.org/level/tileX/tileY.png"
    },
    {
      shapeData: africa_continent,
      type: 'SubLayer',
      shapeSettings: {
        fill: "blue"
      }
    }
  ]
});
maps.appendTo('#element');

```


INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="africa_continent.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Enabling legend

The legend can be added to the tile Maps by setting the [visible](#) property of [legendSettings](#) to **true**.

INDEX.JS

```

var map = new ej.maps.Maps({
  legendSettings: {
    visible: true,
    type: 'Markers',
    useMarkerShape: true,
    toggleLegendSettings: {
      enable: true,
      applyShapeSettings: false,
      border: {
        color: 'green',
        width: 2
      }
    }
  },
  layers: [
    {
      urlTemplate: "https://tile.openstreetmap.org/level/tileX/tileY.png",
      shapeDataPath: 'name',
      shapePropertyPath: 'name',

```

```

        shapeSettings: {
            fill: '#E5E5E5'
        },
        markerSettings: [
            {
                dataSource: markerDataSource,
                colorValuePath: 'color',
                shapeValuePath: 'shape',
                legendText: 'country',
                visible: true
            }
        ]
    }
}
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="markerdata.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Bing maps in EJ2 JavaScript Maps control

Bing Maps is a online Maps provider, owned by Microsoft. As like OSM, it provide Maps tile images based on our requests and combines those images into a single one to display Maps area.

Adding Bing Maps

The Bing Maps can be rendered using the [urlTemplate](#) property, which is based on the URL generated by the [getBingUrlTemplate](#) method in the Maps. The format of the required URL of Bing Maps varies from

other online map providers. As a result, a built-in [getBingUrlTemplate](#) method has been included that returns the URL in a generic format. In the meantime, a subscription key is required for Bing Maps. The Bing Maps key can be obtained from [here](#), then append it to the Bing Maps URL before passing it to the [getBingUrlTemplate](#) method. The URL returned by this method must be passed to the [urlTemplate](#) property.

```
`javascript
var map = new ej.maps.Maps({
load: function(args){
args.maps.getBingUrlTemplate("https://dev.virtualsearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
args.maps.layers[0].urlTemplate= url;
});
},
layers:[{
}]
});
map.appendTo('container');
`
```

Types of Bing Maps

Bing Maps provides different types of Maps and it is supported in the Maps control.

- **Aerial** - Displays satellite images to highlight roads and major landmarks for easy identification.
- **AerialWithLabel** - Displays aerial Maps with labels for the continent, country, ocean, etc.
- **Road** - Displays the default Maps view of roads, buildings, and geography.
- **CanvasDark** - Displays dark version of the road Maps.
- **CanvasLight** - Displays light version of the road Maps.
- **CanvasGray** - Displays grayscale version of the road Maps.

To render the light version of the road Maps, set the **CanvasLight** value is passed via the URL into the [getBingUrlTemplate](#) method demonstrated in the following code sample.

```
`javascript
var map = new ej.maps.Maps({
load: function(args){
args.maps.getBingUrlTemplate("https://dev.virtualsearth.net/REST/V1/Imagery/Metadata/CanvasLight?output=json&uriScheme=https&key=?").then(function(url) {
args.maps.layers[0].urlTemplate= url;
});
},
`
```

```

zoomSettings: {
zoomFactor: 12,
},
centerPosition : {
latitude: 38.8951,
longitude: -77.0364
},
layers:[{
}]
});
map.appendTo('container');
`

```

Enabling zooming and panning

Bing Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a Maps for in-depth analysis. Panning helps to move a Maps around to focus the targeted area.

```

`javascript
var map = new ej.maps.Maps({
load: function(args){
args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
args.maps.layers[0].urlTemplate= url;
});
},
zoomSettings: {
enable: true,
toolBars: [{ "Zoom", "ZoomIn", "ZoomOut", "Pan", "Reset" }]
},
layers: [{
}]
});
map.appendTo('container');
`

```

Adding markers and navigation line

Markers can be added to the layers of Bing Maps by setting the corresponding location's coordinates of latitude and longitude using [markerSettings](#). Navigation lines can be added on top of an Bing Maps layer

for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [navigationLineSettings](#).

```
`javascript
var map = new ej.maps.Maps({
  load: function(args){
    args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
      args.maps.layers[0].urlTemplate= url;
    });
  },
  zoomSettings: {
    zoomFactor: 4
  },
  centerPosition: {
    latitude: 29.394708
    logitude: -94.954653
  },
  layers: [{
    markerSettings: [{
      visible: true,
      height: 25,
      width: 15,
      dataSouce: [{
        latitude: 34.060620,
        longitude: -118.330491,
        name: "California"
      },
      {
        latitude: 40.724546,
        longitude: -73.850344,
        name: "New York"
      }
    ]
  }],
  navigationLineSettings: [{
```

```

visible: true,
color: "blue",
angle: 0.1,
latitude: {
  34.060620, 40.724546
},
longitude: {
  -118.330491, -73.850344
}
}]
}
});
map.appendTo('container');
`

```

Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the Bing Maps layer for highlighting a particular continent or country in Bing Maps by adding another layer and specifying the [type](#) property of Maps layer to **SubLayer**.

```

`javascript
var map = new ej.maps.Maps({
  load: function(args){
    args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
      args.maps.layers[0].urlTemplate= url;
    });
  },
  layers: [
    {
      // Add Bing Map.
    },
    {
      shapeData: Africa_Continent,
      type: 'SubLayer',
      shapeSettings: {

```

```

fill: "blue"
}
}
]
});
map.appendTo('container');
`

```

Enabling legend

The legend can be added to the tile Maps by setting the [visible](#) property of [legendSettings](#) to **true**.

```

`javascript
var map = new ej.maps.Maps({
load: function(args){
args.maps.getBingUrlTemplate("https://dev.virtualsearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
args.maps.layers[0].urlTemplate= url;
});
},
legendSettings: {
visible: true,
type: 'Markers',
useMarkerShape:true,
toggleLegendSettings: {
enable: true,
applyShapeSettings: false,
border: {
color: 'green',
width: 2
}
}
},
layers: [
{
shapeDataPath: 'name',
shapePropertyPath: 'name',

```

```

shapeSettings: {
  fill: '#E5E5E5'
},
markerSettings: [
  {
    dataSource: markerDataSource,
    colorValuePath: 'color',
    shapeValuePath: 'shape',
    legendText: 'country',
    visible: true
  }
]
});
map.appendTo('#element');

```

Azure maps in EJ2 JavaScript Maps control

Azure Maps is yet another online Maps provider, owned by Microsoft. As like OSM and Bing Maps, it provides Maps tile images based on our requests and combines those images into a single one to display Maps area.

Adding Azure Maps

The Azure Maps can be rendered using the [urlTemplate](#) property with the tile server URL provided by online map providers. In the meantime, a subscription key is required for Azure Maps. Follow the steps in this [link](#) to generate an API key, and then added the key to the URL.

Refer to [Azure Maps Licensing](#).

```

`javascript
var map = new ej.maps.Maps({
  layers: [{
    urlTemplate: "https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key &api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY"
  }]
});
map.appendTo('container');

```


Enabling zooming and panning

The Azure Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a map for in-depth analysis. Panning helps to move a map around to focus the targeted area.

```
`javascript
var map = new ej.maps.Maps({
  zoomSettings: {
    enable: true,
    toolbars: ["Zoom", "ZoomIn", "ZoomOut", "Pan", "Reset"]
  },
  layers: [{
    urlTemplate: "https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key &api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY"
  }]
});
map.appendTo('container');
```

Adding markers and navigation line

Markers can be added to the layers of Azure Maps by setting the corresponding location's coordinates of latitude and longitude using [markerSettings](#). Navigation lines can be added on top of the Azure Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [navigationLineSettings](#).

```
`javascript
var map = new ej.maps.Maps({
  zoomSettings: {
    zoomFactor: 4,
  },
  centerPosition: {
    latitude: 29.394708,
    longitude: -94.954653,
  },
  layers: [
    {
      urlTemplate: 'https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key &api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY',
      markerSettings: [
        {
```

```
visible: true,
height: 25,
width: 15,
dataSource: [
{
latitude: 34.06062,
longitude: -118.330491,
name: 'California',
},
{
latitude: 40.724546,
longitude: -73.850344,
name: 'New York',
},
],
navigationLineSettings: [
{
visible: true,
color: 'blue',
width: 5,
angle: 0.1,
latitude: [34.06062, 40.724546],
longitude: [-118.330491, -73.850344],
},
],
});
map.appendTo('container');
```

Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the Azure Maps layer for highlighting a particular continent or country in Azure Maps by adding another layer and specifying the [type](#) property of Maps layer to **SubLayer**.

```
`javascript
var map = new ej.maps.Maps({
  layers: [
    {
      urlTemplate: "https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key &api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY"
    },
    {
      shapeData: Africa_Continent,
      type: 'SubLayer',
      shapeSettings: {
        fill: "blue"
      }
    }
  ]
});
map.appendTo('container');
`
```

Enabling legend

The legend can be added to the Azure Maps by setting the [visible](#) property of [legendSettings](#) to **true**.

```
`javascript
var map = new ej.maps.Maps({
  legendSettings: {
    visible: true,
    type: 'Markers',
    useMarkerShape:true,
    toggleLegendSettings: {
      enable: true,
      applyShapeSettings: false,
      border: {
        color: 'green',
```

```

width: 2
}
}
},
layers: [
{
urlTemplate: "https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key &api-
version=1.0&style=satellite&zoom=level&x=tileX&y=tileY",
shapeDataPath: 'name',
shapePropertyPath: 'name',
shapeSettings: {
fill: '#E5E5E5'
},
markerSettings: [
{
dataSource: markerDataSource,
colorValuePath: 'color',
shapeValuePath: 'shape',
legendText: 'country',
visible: true
}
]
}
]
});
map.appendTo('#element');
`

```

Other maps in EJ2 JavaScript Maps control

Apart from OpenStreetMap and Bing Maps, you can also render Maps from other online map service providers by specifying the URL provided by those providers in the [urlTemplate](#) property. The URL template concept has been implemented in such a way that any online map service providers using the following template can benefit from previewing their Map in the Syncfusion EJ2 Maps control.

<!-- markdownlint-disable MD034 -->

Sample Template: `https://< domain_name >/maps/basic/{z}/{x}/{y}.png`

- "\${z}" - It represents zoom factor (level).
- "\${x}" - It indicates tile image x-position (tileX).
- "\${y}" - It indicates tile image y-position (tileY).

In this case, the key generated for those online map service providers can also be appended to the URL. This allows to create personalized Maps with your own content and imagery.

Following is an example of how to add a TomTom map. You can generate an API key by following the steps in this [link](#) and then adding the key to the URL.

```
`javascript
var map = new ej.maps.Maps({
layers: [{
urlTemplate:
"http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key"
}]
});
map.appendTo('container');
`
```



Enabling zooming and panning

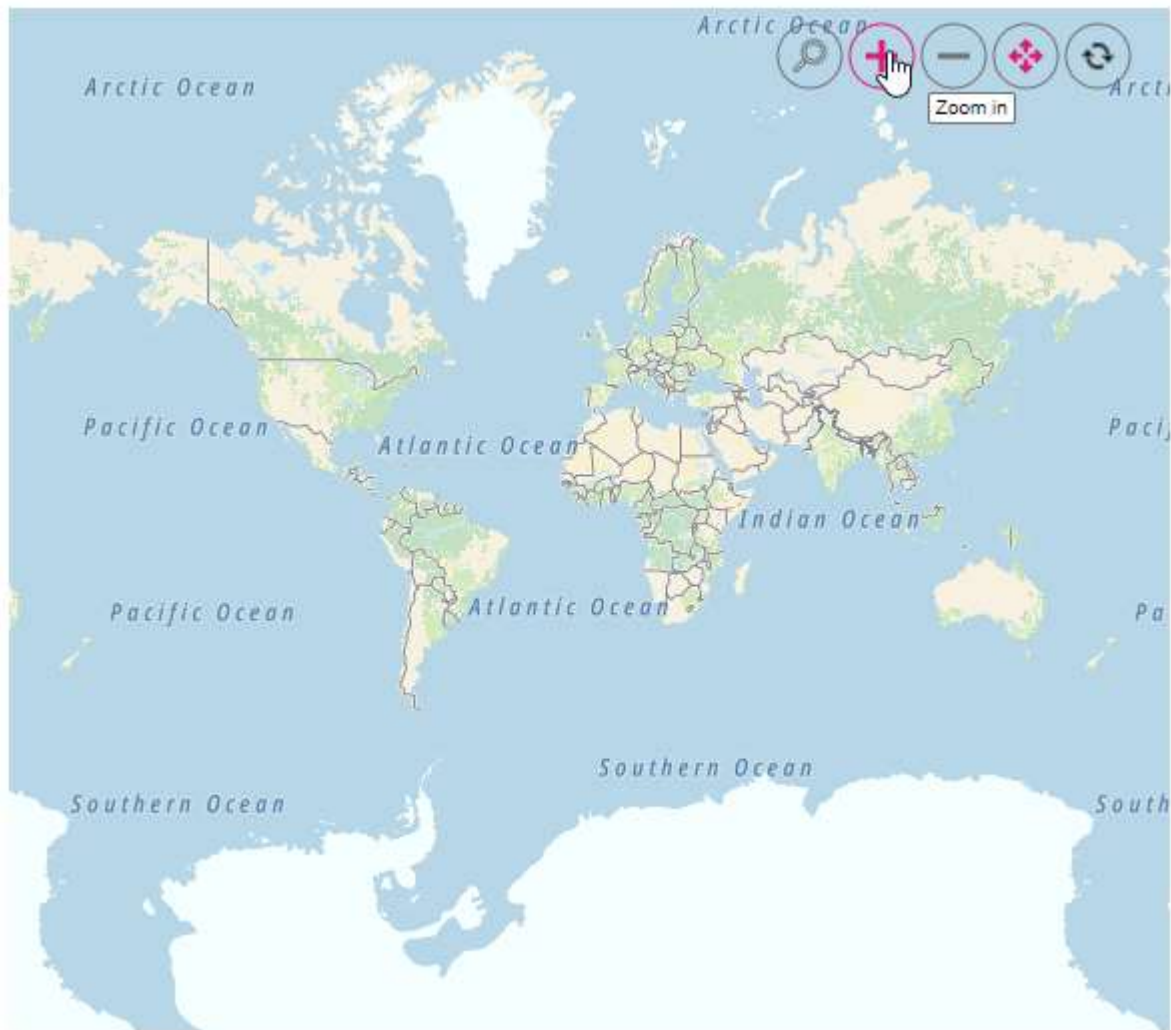
Tile Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a map for in-depth analysis. Panning helps to move a map around to focus the targeted area.

```
`javascript
var maps = new ej.maps.Maps({
  zoomSettings: {
    enable: true,
    toolbars: ["Zoom", "ZoomIn", "ZoomOut", "Pan", "Reset"]
  },
  layers: [
    {
      urlTemplate:
        "http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key"
```

```

}
]
});
maps.appendTo('#element');
`

```



Adding markers and navigation line

Markers can be added to the layers of tile Maps by setting the corresponding location's coordinates of latitude and longitude using [markerSettings](#). Navigation lines can be added on top of an tile Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [navigationLineSettings](#).

```

`javascript
var maps = new ej.maps.Maps({
  zoomSettings: {

```

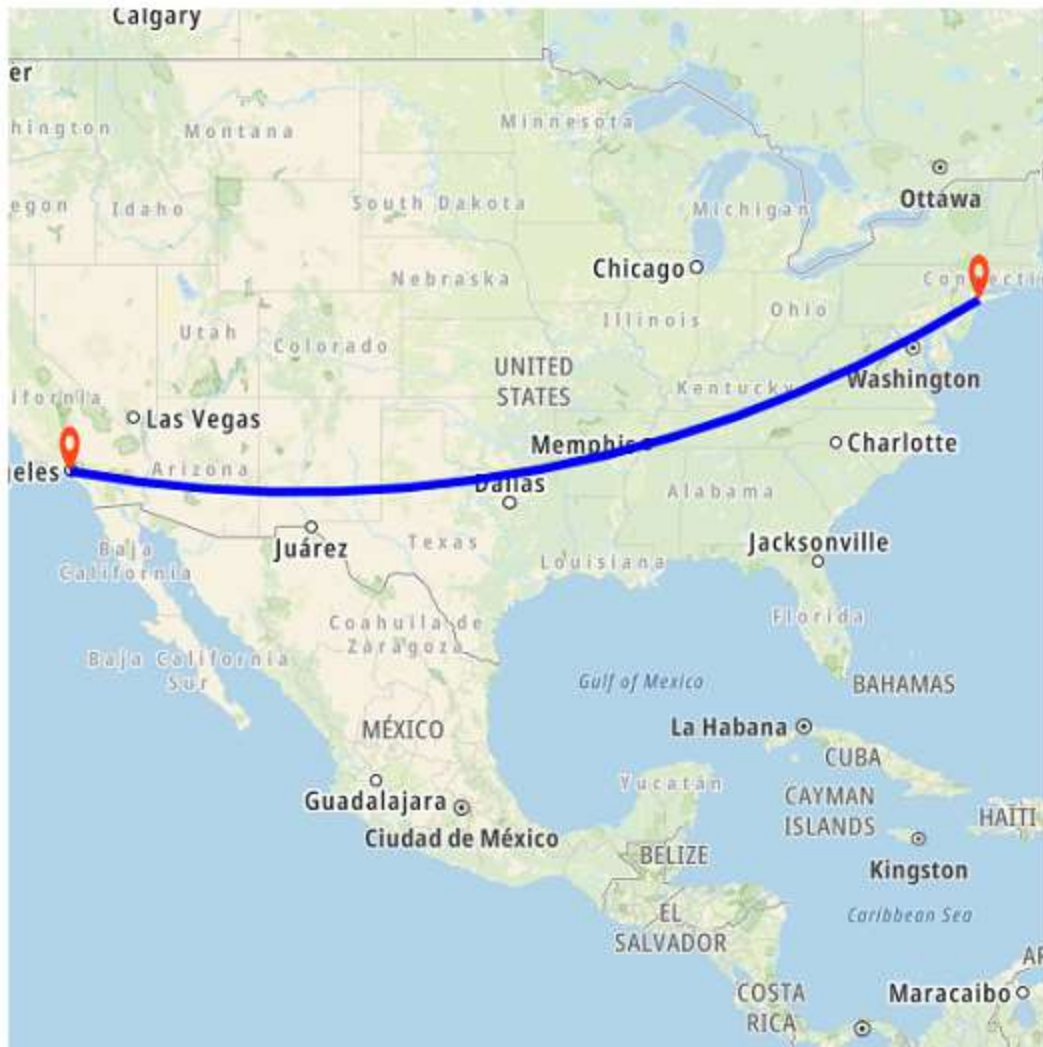
```
zoomFactor: 4
},
centerPosition: {
latitude: 29.394708,
longitude: -94.954653
},
layers: [
{
urlTemplate:
"http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key",
markerSettings: [{
visible: true,
height: 25,
width: 15,
dataSource: [{
latitude: 34.060620,
longitude: -118.330491,
name: "California"
},
{
latitude: 40.724546,
longitude: -73.850344,
name: "New York"
}]
}],
navigationLineSettings: [{
visible: true,
color: "blue",
width: 5,
angle: 0.1,
latitude: [34.060620, 40.724546],
longitude: [-118.330491, -73.850344]
}]
}]
```



```

}
]
});
maps.appendTo('#element');
`

```



Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the tile Maps layer for highlighting a particular continent or country in tile Maps by adding another layer and specifying the [type](#) property of Maps layer to **SubLayer**.

```

`javascript
var maps = new ej.maps.Maps({
layers: [
{

```

```
urlTemplate:
"http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key"
},
{
shapeData: africa_continent,
type: "SubLayer",
shapeSettings: {
fill: "blue"
}
}
]
});
maps.appendTo('#element');
、
```

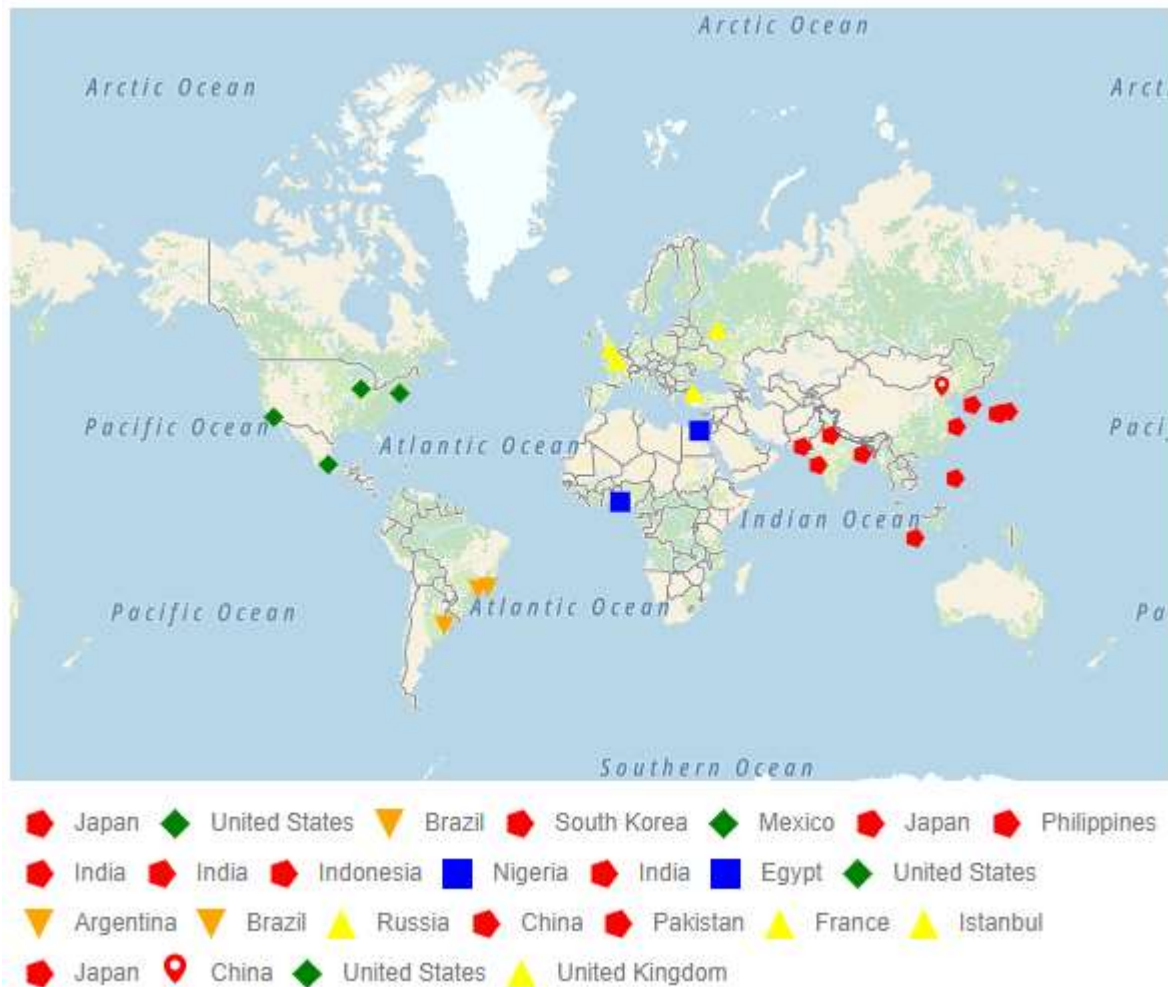


Enabling legend

The legend can be added to the tile Maps by setting the [visible](#) property of [legendSettings](#) to **true**.

```
`javascript
var map = new ej.maps.Maps({
  legendSettings: {
    visible: true,
    type: 'Markers',
    useMarkerShape:true,
    toggleLegendSettings: {
      enable: true,
      applyShapeSettings: false,
      border: {
        color: 'green',
```

```
width: 2
}
},
layers: [
{
urlTemplate:'http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key'
,
shapeDataPath: 'name',
shapePropertyPath: 'name',
shapeSettings: {
fill: '#E5E5E5'
},
markerSettings: [
{
dataSource: markerDataSource,
colorValuePath: 'color',
shapeValuePath:'shape',
legendText: 'country',
visible: true
}
]
}
]
});
map.appendTo('#element');
、
```



Customization in EJ2 JavaScript Maps control

Setting the size for Maps

The width and height of the Maps can be set using the [width](#) and [height](#) properties in the Maps control. Percentage or pixel values can be used for the height and width values.

INDEX.TS

```
import { Maps } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
let map: Maps = new Maps({
  height: '200px',
  width: '500px',
  layers: [{
    shapeData: world_map,
    shapeSettings: {
      autofill: true
    }
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Maps title

The title for the Maps can be set using the [titleSettings](#). It can be customized using the following properties.

- [alignment](#) - To customize the alignment for the text in the title for the Maps. The possible values are **Center**, **Near** and **Far**.
- [description](#) - To set the description of the title in Maps.
- [text](#) - To set the text for the title in Maps.
- [textStyle](#) - To customize the text of the title in Maps.
- [subtitleSettings](#) - To customize the subtitle for the Maps.

INDEX.TS

```

import { Maps } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
let map: Maps = new Maps({
  titleSettings: {
    text: 'Maps Control',
    textStyle: {
      color: 'red',
      fontStyle: 'italic',
      fontWeight: 'regular',
      fontFamily: 'arial',

```

```

        size: '14px'
    },
    alignment: 'Center'
},
layers: [{
    shapeData: world_map,
    shapeSettings: {
        autofill: true,
    }
}]
});
map.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Setting theme

The Maps control supports following themes.

- Material
- Fabric
- Bootstrap
- Highcontrast
- MaterialDark
- FabricDark

- BootstrapDark
- Bootstrap4
- HighContrastLight
- Tailwind

By default, the Maps are rendered by the **Material** theme. The theme of the Maps component is changed using the [theme](#) property.

INDEX.TS

```
import { Maps, MapsTheme } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
let map: Maps = new Maps({
  layers: [{
    shapeData: world_map,
    shapeSettings: {
      autofill: true
    }
  }]
});
map.appendTo('#container');
document.getElementById('theme').onchange = () => {
  var value = (<HTMLInputElement>document.getElementById('theme')).value;
  map.theme= <MapsTheme>value;
  map.refresh();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div class="col-lg-9 control-section">
    <div id="container" align="center"></div>
  </div>
  <div class="col-lg-3 property-section">
    <table id="property" title="Properties" style="width: 100%">
      <tbody><tr style="height: 50px">
        <td style="width: 60%">
          <div>Projection Type</div>
        </td>
        <td style="width: 40%;">
```



```

        <select name="theme" id="theme" style="margin-left: -
25px">
            <option value="Material">Material</option>
            <option value="Fabric">Fabric</option>
            <option value="Bootstrap">Bootstrap</option>
            <option value="Highcontrast">Highcontrast</option>
            <option value="MaterialDark">MaterialDark</option>
            <option value="FabricDark">FabricDark</option>
            <option value="BootstrapDark">BootstrapDark</option>
            <option value="Bootstrap4">Bootstrap4</option>
        </select>
    </td>
</tr>
</tbody></table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing Maps container

The following properties are available to customize the container in the Maps.

- [background](#) - To apply the background color to the container in the Maps.
- [border](#) - To customize the color, width and opacity of the border of the Maps.
- [margin](#) - To customize the margins of the Maps.

INDEX.TS

```

import { Maps } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
let map: Maps = new Maps({
    background: '#CCD1D1',
    border: {
        color: 'green',
        width: 2
    },
    margin: {
        bottom: 10,
        left: 20,
        right: 20,
        top: 10
    },
    layers: [{
        shapeData: world_map,
        shapeSettings: {
            autofill: true
        }
    }]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Customizing Maps area

By default, the background color of the shape maps is set as **white**. To modify the background color of the Maps area, the [background](#) property in the [mapsArea](#) is used. The border of the Maps area can be customized using the [border](#) property in the [mapsArea](#).

INDEX.TS

```

import { Maps } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
let map: Maps = new Maps({
  mapsArea: {
    background: '#CCD1D1',
    border: {
      width: 2,
      color: 'green'
    }
  },
  layers: [{
    shapeData: world_map,
    shapeSettings: {
      autofill: true
    }
  }]
});

```

```
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Customizing the shapes

The following properties are available in [shapeSettings](#) to customize the shapes of the Maps.

- [fill](#) - To apply the fill color to the all the shapes.
- [autofill](#) - To apply the palette colors to the shapes if it is set as true.
- [palette](#) - To set the custom palette for the shapes.
- [border](#) - To customize the color, width and opacity of the border of the shapes.
- [dashArray](#) - To define the pattern of dashes and gaps that is applied to the outline of the shapes.
- [opacity](#) - To customize the transparency for the shapes.

INDEX.TS

```
import { Maps } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
let map: Maps = new Maps({
  layers: [{
    shapeData: world_map,
    shapeSettings: {
      autofill: true,
```

```

        palette: ['#C7DE6C', '#59A076', '#88D0BC', '#FEA78C',
        '#FFC557'],
        border: {
            color: '#FEE1DD',
            width: 3
        },
        dashArray: '1',
        opacity: 0.9
    }
}
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Setting color to the shapes from the data source

The color for each shape in the Maps can be set using the [colorValuePath](#) property of [shapeSettings](#). The value for the [colorValuePath](#) property is the field name from the data source of the [shapeSettings](#) which contains the color values.

INDEX.TS

```

import { Maps } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
let map: Maps = new Maps({
    layers: [{

```

```

    shapeData: world_map,
    shapePropertyPath: 'continent',
    shapeDataPath: 'continent',
    dataSource: [
        { continent: "North America", color: '#71B081' },
        { continent: "South America", color: '#5A9A77' },
        { continent: "Africa", color: '#498770' },
        { continent: "Europe", color: '#39776C' },
        { continent: "Asia", color: '#266665' },
        { continent: "Oceania", color: '#124F5E' }
    ],
    shapeSettings: {
        colorValuePath: 'color'
    }
  }
}
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Applying border to individual shapes

The border of each shape in the Maps can be customized using the [borderColorValuePath](#) and [borderWidthValuePath](#) properties to modify the color and the width of the border respectively. The field name in the data source of the layer which contains the color and the width values must be set in the [borderColorValuePath](#) and [borderWidthValuePath](#) properties respectively. If the values of

[borderWidthValuePath](#) and [borderColorValuePath](#) do not match with the field name from the data source, then the color and width of the border will be applied to the shapes using the [border](#) property in the [shapeSettings](#).

INDEX.TS

```
import { Maps } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
let map: Maps = new Maps({
  layers: [{
    shapeData: world_map,
    shapePropertyPath: 'continent',
    shapeDataPath: 'continent',
    dataSource: [
      { continent: "North America", color: '#71B081', borderColor:
'#CCFFE5', width: 2 },
      { continent: "South America", color: '#5A9A77', borderColor:
'red', width: 2 },
      { continent: "Africa", color: '#498770', borderColor: '#FFCC99',
width: 2 },
      { continent: "Europe", color: '#39776C', borderColor: '#66B2FF',
width: 2 },
      { continent: "Asia", color: '#266665', borderColor: '#999900',
width: 2 },
      { continent: "Oceania", color: '#124F5E', borderColor: 'blue',
width: 2 }
    ],
    shapeSettings: {
      borderColorValuePath: 'borderColor',
      borderWidthValuePath: 'width',
      colorValuePath: 'color'
    }
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
```

```

</div>
<script>
    var ele = document.getElementById('container');
    if (ele) {
        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Projection type

The Maps control supports the following projection types:

- Mercator
- Equirectangular
- Miller
- Eckert3
- Eckert5
- Eckert6
- Winkel3
- AitOff

By default, the Maps are rendered by the **Mercator** projection type in which the Maps are rendered based on the coordinates. So, the Maps is not stretched. To change the type of projection in the Maps, the [projectionType](#) property is used.

INDEX.TS

```

import { Maps, ProjectionType } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
import { projectionData } from './projection-data.ts';
let map: Maps = new Maps({
    projectionType: 'Mercator',
    layers: [{
        shapeData: world_map,
        shapeDataPath: 'Country',
        shapePropertyPath: 'name',
        dataSource: projectionData,
        tooltipSettings: {
            visible: true,
            valuePath: 'Country'
        },
    },
    shapeSettings: {
        fill: '#E5E5E5',
        colorMapping: [
            {
                value: 'Permanent',
                color: '#EDB46F'
            },
            {
                color: '#F1931B',
                value: 'Non-Permanent'
            }
        ]
    }
}

```

```

        ],
        colorValuePath: 'Membership'
    }
}
});
map.appendTo('#container');
document.getElementById('projectiontype').onchange = function(){
    let ele: HTMLSelectElement =
    (<HTMLSelectElement>document.getElementById('projectiontype'))
    maps.projectionType = <ProjectionType>ele.value;
    maps.refresh();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="projection-data.js"></script>
    <script src="world-map.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="col-lg-9 control-section">
        <div id="container" align="center"></div>
    </div>
    <div class="col-lg-3 property-section">
        <table id="property" title="Properties" style="width: 100%">
            <tbody><tr style="height: 50px">
                <td style="width: 60%">
                    <div>Projection Type</div>
                </td>
                <td style="width: 40%;">
                    <select name="projectionType" id="projectiontype"
style="margin-left: -25px">
                        <option value="Mercator">Mercator</option>
                        <option
value="Equirectangular">Equirectangular</option>
                        <option value="Miller">Miller</option>
                        <option value="Eckert3">Eckert3</option>
                        <option value="Eckert5">Eckert5</option>
                        <option value="Eckert6">Eckert6</option>
                        <option value="Winkel3">Winkel3</option>
                        <option value="AitOff">AitOff</option>
                    </select>
                </td>
            </tr>
        </tbody>
    </table>

```



```

        </tr>
      </tbody></table>
    </div>

    <script>
    var ele = document.getElementById('container');
    if(ele) {
      ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
  </body></html>

```

Color mapping in EJ2 JavaScript Maps control

Color mapping is used to customize the shape colors based on the given values. It has three types: range color mapping, equal color mapping and desaturation color mapping. To add color mapping to the shapes of the Maps, bind the data source to the [dataSource](#) property of [layerSettings](#) and set the field name which contains the color value in the data source to the [colorValuePath](#) property.

Range color mapping

Range color mapping applies the color to the shapes of the Maps which matches the numeric values in the data source within the given color mapping ranges. The [from](#) and [to](#) properties in the [colorMapping](#) are used to specify the color mapping ranges in the Maps.

```
`ts
```

```
export var Population_Density = [
```

```
...
```

```
{
```

```
'code': 'AE',
```

```
'value': 90,
```

```
'name': 'United Arab Emirates',
```

```
'population': 8264070,
```

```
'density': 99
```

```
},
```

```
{
```

```
'code': 'GB',
```

```
'value': 257,
```

```
'name': 'United Kingdom',
```

```
'population': 62041708,
```

```
'density': 255
```

```
},
```

```
{
```

```

'code': 'US',
'value': 34,
'name': 'United States',
'population': 325020000,
'density': 33
}
...
];
`

```

Bind the **Population_Density** data to the [dataSource](#) property of [layerSettings](#) and set the [colorValuePath](#) property of [shapeSettings](#) as **density**. The range values can be set using the [from](#) and [to](#) properties of [colorMapping](#).

INDEX.JS

```

var map = new ej.maps.Maps({
  legendSettings:{
    visible:true,
  },
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    dataSource: Population_Density,
    shapeSettings: {
      colorValuePath: 'density',
      fill: '#E5E5E5',
      colorMapping: [
        {
          from: 0.00001, to: 100, color: 'rgb(153,174,214)'
        },
        {
          from: 100, to: 200, color: 'rgb(115,143,199)'
        },
        {
          from: 200, to: 300, color: 'rgb(77,112,184)'
        },
        {
          from: 300, to: 500, color: 'rgb(38,82,168)'
        },
        {
          from: 500, to: 19000, color: 'rgb(0,51,153)'
        }
      ]
    }
  }
]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="data.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Equal color mapping

Equal color mapping applies the color to the shapes of the Maps when the [value](#) property of [colorMapping](#) matches with the values provided in the data source. The following example shows how to apply equal color mapping to the shapes with the data source **unCountries** which illustrates the permanent and non-permanent countries in the UN security council.

```
`ts
```

```

export var unCountries: object[] = [
  { Country: 'China', Membership: 'Permanent' },
  { Country: 'France', Membership: 'Permanent' },
  { Country: 'Russia', Membership: 'Permanent' },
  { Country: 'United Kingdom', Membership: 'Permanent' },
  { Country: 'United States', Membership: 'Permanent' },
  { Country: 'Bolivia', Membership: 'Non-Permanent' },
  { Country: 'Eq. Guinea', Membership: 'Non-Permanent' },
  { Country: 'Ethiopia', Membership: 'Non-Permanent' },

```

```
{ Country: "Côte d'Ivoire", Membership: 'Permanent' },
{ Country: 'Kazakhstan', Membership: 'Non-Permanent' },
{ Country: 'Kuwait', Membership: 'Non-Permanent' },
{ Country: 'Netherlands', Membership: 'Non-Permanent' },
{ Country: 'Peru', Membership: 'Non-Permanent' },
{ Country: 'Poland', Membership: 'Non-Permanent' },
{ Country: 'Sweden', Membership: 'Non-Permanent' },
];
`
```

Bind the **unCountries** data to the [dataSource](#) property of [layerSettings](#) and set the [colorValuePath](#) property of [shapeSettings](#) as **Membership**. Set the [value](#) property in the [colorMapping](#) to **Permanent** and **Non-Permanent** in the different set of color mapping properties. If the corresponding value of the [colorValuePath](#) property matches with the corresponding field name in the data source, then the given color will be applied.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'Country',
    shapePropertyPath: 'name',
    dataSource : unCountries,
    shapeSettings: {
      fill: '#E5E5E5',
      colorMapping: [
        {
          value: 'Permanent',
          color: '#C3E6ED'
        },
        {
          color: '#F1931B',
          value: 'Non-Permanent'
        }
      ]
    },
    colorValuePath: 'Membership'
  }
]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<script src="world-map.js"></script>
<script src="data.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Desaturation color mapping

Desaturation color mapping applies the color to the shapes of the Maps, similar to the range color mapping. The opacity will be applied in this color mapping based on the [minOpacity](#) and [maxOpacity](#) properties in the [colorMapping](#).

The following example shows how to apply desaturation color mapping to the shapes with the data source **Population_Density** that is available in the [Range color mapping](#) section. Bind the **Population_Density** data to the [dataSource](#) property of [layerSettings](#) and set the [colorValuePath](#) property of [shapeSettings](#) as **density**. The range values can be set using the [from](#) and [to](#) properties of [colorMapping](#).

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    dataSource: Population_Density,
    legendSettings: {
      visible: true
    },
    shapeSettings: {
      colorValuePath: 'density',
      colorMapping: [
        {
          from: 0, to: 100, color: 'red', minOpacity: 0.2,
maxOpacity: 1
        },
        {

```

```

        from: 101, to: 200, color: 'blue', minOpacity:0.3,
maxOpacity:1
    }
    ]
}
}]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="data.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Multiple colors for a single shape

Multiple colors can be added to the color mapping which can be used as gradient effect to a specific shape based on the ranges in the data source. By using the [color](#) property of [colorMapping](#), any number of colors can be set to the shapes as a gradient.

The following example demonstrates how to use multiple colors in color mapping with the data source **Population_Density** that is available in the [Range color mapping](#) section. Bind the **Population_Density** data to the [dataSource](#) property of [layerSettings](#) and set the [colorValuePath](#) property of [shapeSettings](#) as **density**. The range values can be set using the [from](#) and [to](#) properties of [colorMapping](#).

INDEX.JS

```

var map = new ej.maps.Maps({
    layers: [{

```

```

        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        dataSource: Population_Density,
        legendSettings: {
            visible: true
        },
        shapeSettings: {
            colorValuePath: 'density',
            colorMapping: [
                {
                    from: 0, to: 100, color: ['red', 'blue']
                },
                {
                    from: 101, to: 200, color: ['green', 'yellow']
                }
            ]
        }
    }
}
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="data.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Color for items excluded from color mapping

Color mapping can be applied to the shapes in the Maps which does not match color mapping criteria such as range or equal values using the [color](#) property of [colorMapping](#).

The following example shows how to set the color for items excluded from the color mapping with the data source **Population_Density** that is available in the [Range color mapping](#) section. In the following example, color mapping is added for the ranges from 0 to 200. If there are any records in the data source that are outside of this range, the color mapping will not be applied. To apply the color for these excluded items, set the [color](#) property alone in the [colorMapping](#).

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    dataSource: Population_Density,
    legendSettings: {
      visible: true
    },
    shapeSettings: {
      colorValuePath: 'density',
      colorMapping: [
        {
          from: 0, to: 100, color: 'skyblue',
        },
        {
          from: 101, to: 200, color: 'blue',
        },
        {
          color: 'green'
        }
      ]
    }
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="data.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
```



```

</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Color mapping for bubbles

The color mapping types such as range color mapping, equal color mapping and desaturation color mapping are applicable for bubbles in the Maps. To add color mapping for bubbles of the Maps, bind the data source to the [dataSource](#) property of [bubbleSettings](#) and set the field name which contains the color value in the data source to the [colorValuePath](#) property. Multiple colors for a single set of bubbles and color for excluded items from [colorMapping](#) are also applicable for bubbles.

INDEX.JS

```

var map = new ej.maps.Maps({
  legendSettings:{
    visible:true,
  },
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    bubbleSettings: [{
      visible: true,
      minRadius: 5,
      dataSource: [
        { name: 'India', population: '38332521' },
        { name: 'New Zealand', population: '19651127' },
        { name: 'Pakistan', population: '3090416' }
      ],
      fill: 'green',
      valuePath: 'population',
      colorValuePath: 'population',
      colorMapping: [{
        value: '38332521',
        color: '#C3E6ED'
      },
      {
        value: '19651127',
        color: '#F1931B'
      },
      {
        value: '3090416',
        color: 'blue'
      }
    ]
  }
]

```

```

    }]
  }
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Data label in EJ2 JavaScript Maps control

Data labels provide information to users about the shapes of the Maps control. It can be enabled by setting the [visible](#) property of the [dataLabelSettings](#) to **true**.

Adding data labels

To display the data labels in the Maps, set the field name containing the text to be displayed from the data source or shape data in the [labelPath](#) property of the [dataLabelSettings](#) property.

In the following example, the value of [labelPath](#) property is set from the field name in the shape data of the Maps layer.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [
    {
      shapeData: usa_map,
      shapeSettings: {
        autofill: true

```

```

        },
        dataLabelSettings: {
            visible: true,
            labelPath: 'name',
        }
    }
}
]);
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="usa.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

In the following example, the value of `labelPath` property is set from the field name in the data source of the layer settings.

INDEX.JS

```

var map = new ej.maps.Maps({
    layers: [{
        shapeData: world_map,
        shapePropertyPath: 'name',
        shapeDataPath: 'name',
        dataLabelSettings: {
            visible: true,
            labelPath: "continent",
            smartLabelMode: 'Trim'
        }
    },

```

```

        dataSource: [
            { "name": "Afghanistan", "value": 53, "countryCode": "AF",
"population": "29863010", "color": "red", "density": "119", "continent":
"Asia" },
            { "name": "Albania", "value": 117, "countryCode": "AL",
"population": "3195000", "color": "Blue", "density": "111", "continent":
"Europe" },
            { "name": "Algeria", "value": 15, "countryCode": "DZ",
"population": "34895000", "color": "Green", "density": "15", "continent":
"Africa" }
        ],
        shapeSettings: {
            autofill: true
        }
    }
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Customization

The following properties are available in the `dataLabelSettings` to customize the data label of the Maps control.

- [border](#) - To customize the color, width and opacity for the border of the data labels in Maps.

- [fill](#) - To apply the color of the data labels in Maps.
- [opacity](#) - To customize the transparency of the data labels in Maps.
- [textStyle](#) - To customize the text style of the data labels in Maps.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [
    {
      shapeData: usa_map,
      shapeSettings: {
        autofill: true
      },
      dataLabelSettings: {
        visible: true,
        labelPath: 'name',
        smartLabelMode: 'Hide',
        intersectionAction: 'Trim',
        border: {
          color: 'green',
          width: 2
        },
        fill: 'transparent',
        opacity: 0.9,
        textStyle: {
          size: '17px',
          fontStyle: 'Sans-serif',
          fontWeight: 'normal'
        }
      },
      tooltipSettings: {
        visible: true,
        valuePath: 'name'
      }
    }
  ]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="usa.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Label animation

The data labels can be animated during the initial rendering of the Maps. This can be enabled by setting the [animationDuration](#) property in the [dataLabelSettings](#) of the Maps. The duration of the animation is specified in milliseconds.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: usa_map,
    dataLabelSettings: {
      visible: true,
      labelPath: 'name',
      smartLabelMode: 'Hide',
      intersectionAction: 'Trim',
      animationDuration: 2000,
    },
    shapeSettings: {
      autofill: true,
    },
    tooltipSettings: {
      visible: true,
      valuePath: 'name',
    },
  ]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="usa.js"></script>

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Smart labels

The Maps control provides an option to handle the labels when they intersect with the corresponding shape borders using the [smartLabelMode](#) property. The following options are available in the `smartLabelMode` property.

- **None** - It specifies that no action is taken, when a label exceeds the shape's region.
- **Hide** - It specifies to hide the labels, when it exceeds the shape's region.
- **Trim** - It specifies to trim the labels, when it exceeds the shape's region.

INDEX.JS

```

var map = new ej.maps.Maps({
    layers: [
        {
            shapeData: usa_map,
            shapeSettings: {
                autofill: true
            },
            dataLabelSettings: {
                visible: true,
                labelPath: 'name',
                smartLabelMode: 'Hide'
            }
        }
    ]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<script src="usa.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container" style="height: 500px; width: 700px">
  <div id="element"></div>
</div>
<script>
  var ele = document.getElementById('container');
  if (ele) {
    ele.style.visibility = "visible";
  }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Intersect action

The Maps component provides an option to handle the labels when a label intersects with another label using the [intersectionAction](#) property. The following options are available in the [intersectionAction](#) property.

- **None** - It specifies that no action is taken, when the labels intersect.
- **Hide** - It specifies to hide the labels when they intersect.
- **Trim** - It specifies to trim the labels when they intersect.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [
    {
      shapeData: usa_map,
      shapeSettings: {
        autofill: true
      },
      dataLabelSettings: {
        visible: true,
        labelPath: 'name',
        intersectionAction: 'Trim'
      }
    }
  ]
});
map.appendTo('#element');

```

INDEX.HTML

```
<!DOCTYPE html>
```



```

<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="usa.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Adding data label as a template

Any HTML elements can be added as a template in the data labels by using the [template](#) property of `dataLabelSettings` in the Maps control.

The properties of data label such as, `smartLabelMode`, `intersectionAction`, `animationDuration`, `border`, `fill`, `opacity` and `textStyle` properties are not applicable to `template` property. The styles can be applied to the label template using the CSS styles of the HTML element.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [
    {
      shapeData: usa_map,
      shapeDataPath: 'Name',
      shapePropertyPath: 'name',
      dataSource: [
        { Name: 'Iowa', Population: 29863010 },
        { Name: 'Utah', Population: 1263010 },
        { Name: 'Texas', Population: 963010 }
      ],
      shapeSettings: {
        autofill: true
      },
      dataLabelSettings: {
        visible: true,
        labelPath: 'Name',

```

```

        template: '<div><div> </div> ${Name}</img></div>'
    }
}
]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="usa.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Polygon shape in EJ2 JavaScript Maps control

The Maps control allows you to add polygon shape to a geometry map or an online map by using the properties in the [polygons](#). This section describes how to add polygon shape to the map and customize them.

The polygon shape can be rendered over the map layer by defining the [points](#) property in the **polygons** of the Maps control. The **points** property uses a collection of latitude and longitude values to define the polygon shape.

The **polygons** provides the following properties for customizing the polygon shape of the Maps control.

- [fill](#) - It is used to change the color of the polygon shape.
- [opacity](#) - It is used to change the opacity of the polygon shape.
- [borderColor](#) - It is used to change the color of the border in the polygon shape.

- [borderWidth](#) - It is used to change the width of the border in the polygon shape.
- [borderOpacity](#) - It is used to change the opacity of the border in the polygon shape.

You can also include “n” polygon shapes using the [polygons](#) property.

The following example shows how to customize the polygon shape over the geometry map.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [
    {
      shapeData: world_map,
      polygonSettings: {
        polygons: [
          {
            points: [
              { longitude: 34.88539587371454, latitude:
28.181421087099537 },
              { longitude: 37.50029619722466, latitude:
24.299419888989462 },
              { longitude: 39.22241423764024, latitude:
22.638529461838658 },
              { longitude: 38.95650769309776, latitude:
21.424998160017495 },
              { longitude: 40.19963938650778, latitude:
20.271205391339606 },
              { longitude: 41.76547269134551, latitude:
18.315451049867193 },
              { longitude: 42.78452077838921, latitude:
16.097235052947966 },
              { longitude: 43.36984949591576, latitude:
17.188572054533054 },
              { longitude: 44.12558191797012, latitude:
17.407258102232234 },
              { longitude: 46.69027032797584, latitude:
17.33342243475734 },
              { longitude: 47.09312386141585, latitude:
16.97087769526452 },
              { longitude: 48.3417299826302, latitude:
18.152700711188004 },
              { longitude: 49.74762591400318, latitude:
18.81544363931681 },
              { longitude: 52.41428026336621, latitude:
18.9035706497573 },
              { longitude: 55.272683129240335, latitude:
20.133861012918544 },
              { longitude: 55.60121336079203, latitude:
21.92042703112351 },
              { longitude: 55.08204399107967, latitude:
22.823302662258882 },
              { longitude: 52.743894337844154, latitude:
22.954463486477437 },
              { longitude: 51.47035908651375, latitude:
24.35818837668566 },
              { longitude: 51.122553219055874, latitude:
24.666679732426346 },
            ]
          }
        ]
      }
    }
  ]
});
```

```

25.173806925822717 },
25.84556484481108 },
26.168494193631147 },
25.349051242147596 },
24.779242606720743 },
25.66825106363693 },
26.268905608606616 },
27.15116474192905 },
27.55738830340198 },
28.566207269716173 },
28.5938991332588 },
29.009321449856984 },
29.157358362696385 },
31.23489959729713 },
31.9642352513131 },
32.18348471414393 },
31.47710220862595 },
30.4851028633376 },
30.3636358598429 },
29.960155516804974 },
29.882136586478993 },
29.15308642012721 },
29.3103032832622 },
28.135787235699823 },
28.181421087099537 }

],
fill: 'red',
opacity: 0.7,
borderColor: 'green',
borderWidth: 2,
borderOpacity: 0.7
}

```

```

    }
  }
]
});
map.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Markers in EJ2 JavaScript Maps control

Markers are notes that are used to leave a message on the Maps. It indicates or marks a specific location with desired symbols on the Maps. It can be enabled by setting the [visible](#) property of the [markerSettings](#) to **true**.

Adding marker

The [dataSource](#) property of the [markerSettings](#) has a list of objects that contains the data for Markers. Using this property, any number of markers can be added to the shape layers. By default, it displays the markers based on the specified latitude and longitude in the given data source. Each data source object contains the following list of properties.

- label - Text that displays some information about the marker in text format.
- latitude - The latitude point which determines the X location of the marker.
- longitude - The longitude point which determines the Y location of the marker.

markerSettings is an Array property.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    markerSettings: [
      {
        visible: true,
        height: 20,
        width: 20,
        dataSource: [
          { latitude: 49.95121990866204, longitude:
18.468749999999998 },
          { latitude: 59.88893689676585, longitude: -109.3359375
},
          { latitude: -6.64607562172573, longitude: -
55.546874999999999 }
        ],
        animationDuration: 0,
      },
    ]
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Adding marker template

The Marker can be added as a template in the Maps component. The [template](#) property of the [markerSettings](#) is used to set the Marker as a template. HTML element or id of an element can be added as the template in Markers.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    markerSettings: [
      {
        visible: true,
        template: '<div id="marker4" class="markerTemplate">Europe' +
          '</div>',
        dataSource: [
          { latitude: 49.95121990866204, longitude: 18.468749999999998 }
        ]
      },
      {
        visible: true,
        template: '<div id="marker5" class="markerTemplate" style="width:50px">North America' +
          '</div>',
        dataSource: [
          { latitude: 59.88893689676585, longitude: -109.3359375 }
        ],
        animationDuration: 0
      },
      {
        visible: true,
        template: '<div id="marker6" class="markerTemplate" style="width:50px">South America' +
          '</div>',
        dataSource: [
          { latitude: -6.64607562172573, longitude: -55.54687499999999 }
        ],
        animationDuration: 0
      }
    ]
  }
]);
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<script src="world-map.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Customization

The following properties are available in [markerSettings](#) to customize the Markers of the Maps component.

- [border](#) - To customize the color, width and opacity of the border for the markers in Maps.
- [fill](#) - To apply the color for markers in Maps.
- [dashArray](#) - To define the pattern of dashes and gaps that is applied to the outline of the markers in Maps.
- [height](#) - To customize the height of the markers in Maps.
- [width](#) - To customize the width of the markers in Maps.
- [offset](#) - To customize the position of the markers in Maps.
- [opacity](#) - To customize the transparency of the markers in Maps.
- [animationDelay](#) - To change the time delay in the transition for markers.
- [animationDuration](#) - To change the time duration of animation for markers.
- [highlightSettings](#) - To customize the highlight settings for the marker in Maps.
- [selectionSettings](#) - To customize the selection settings for the marker in Maps.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    markerSettings: [{
      visible: true,
      border: {
        color: 'green',
        width: 2
      },
      fill: 'red',

```



```

        dashArray: '1',
        height: 20,
        width: 20,
        opacity: 0.9,
        animationDelay: 100,
        animationDuration: 1000,
        shape: 'Balloon',
        dataSource: [
            { latitude: 37.0000, longitude: -120.0000, city:
'California' },
            { latitude: 40.7127, longitude: -74.0059, city: 'New York'
},
            { latitude: 42, longitude: -93, city: 'Iowa' }
        ]
    }
}
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Marker shapes

The Maps component supports the following marker shapes. To set the shape of the marker, the [shape](#) property in [markerSettings](#) is used.

- Balloon
- Circle
- Cross
- Diamond
- Image
- Rectangle
- Start
- Triangle
- VerticalLine
- HorizontalLine

Rendering Marker shape as image

To render a marker as an image in Maps, set the [shape](#) property of [markerSettings](#) as **Image** and specify the path of the image to [imageUrl](#) property. There is another way to render a marker as an image using the [imageUrlValuePath](#) property of the [markerSettings](#). Bind the field name that contains the path of the image in the data source to the [imageUrlValuePath](#) property.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    markerSettings: [{
      visible: true,
      shape: 'Image',
      imageUrl:
'https://ej2.syncfusion.com/javascript/demos/src/maps/images/ballon.png',
      height: 20,
      width: 20,
      dataSource: [
        { latitude: 37.0000, longitude: -120.0000, city:
'California' },
        { latitude: 40.7127, longitude: -74.0059, city: 'New York' },
        { latitude: 42, longitude: -93, city: 'Iowa' }
      ]
    }]
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Multiple marker groups

Multiple group of markers can be added to the Maps using the [markerSettings](#) in which the properties of markers are added as an array. The customization for the markers can be done with the [markerSettings](#).

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    markerSettings: [{
      visible: true,
      shape: 'Diamond',
      height: 15,
      fill: "green",
      width: 15,
      dataSource: [
        {
          latitude: 37.0000, longitude: -120.0000,
          name: 'California',
        },
        {
          latitude: 40.7127, longitude: -74.0059, name: "New York"
        },
        {
          latitude: 42, longitude: -93, name: 'Iowa'
        }
      ]
    },
    {
      visible: true,
      dataSource: [{
        latitude: 19.228825, longitude: 72.854118, name: "Mumbai"
      }, {
        latitude: 28.610001, longitude: 77.230003, name: "Delhi"
      }, {
        latitude: 13.067439, longitude: 80.237617, name: "Chennai"
```

```

        },
        shape: 'Circle',
        fill: "blue",
        height: 10,
        width: 10
    }
  ]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Customize marker shapes from data source

Bind different colors and shapes to the marker from data source

Using the [shapeValuePath](#) and [colorValuePath](#) properties, apply the color and shape of the marker from the given data source. Bind the data source to the [dataSource](#) property of the [markerSettings](#) and set the field names that contains the shape and color values in the data source to the [shapeValuePath](#) and [colorValuePath](#) properties. A default marker object is represented by a **balloon** shape. To set various shapes to the marker object by using the [shape](#) property in [markerSettings](#).

The following shapes are used for the marker object.

- Circle
- Rectangle

- Balloon
- Cross
- Polyline
- Diamond
- Star
- Triangle
- HorizontalLine
- VerticalLine
- pentagon

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    markerSettings: [{
      visible: true,
      shapeValuePath: 'shape',
      colorValuePath: 'color',
      dataSource: [
        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe', color: 'red', shape: 'Triangle' },
        { latitude: 59.88893689676585, longitude: -109.3359375,
name: 'North America',
        color: 'blue', shape: 'Pentagon' },
        { latitude: -6.64607562172573, longitude: -
55.54687499999999, name: 'South America',
        color: 'green', shape: 'InvertedTriangle' }
      ]
    }]
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="world-map.js"></script>

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

<div id="container" style="height: 500px; width: 700px">
  <div id="element"></div>
</div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Setting value path from the data source

The latitude and longitude values are used to determine the location of each marker in the Maps. The [latitudeValuePath](#) and [longitudeValuePath](#) properties are used to specify the value path that presents in the data source of the marker. In the following example, the field name from the data source is set to the [latitudeValuePath](#) and [longitudeValuePath](#) properties.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    markerSettings: [{
      visible: true,
      latitudeValuePath: 'latitude',
      longitudeValuePath: 'longitude',
      dataSource: [
        { latitude: 49.95121990866204, longitude: 18.468749999999998 },
        { latitude: 59.88893689676585, longitude: -109.3359375 },
        { latitude: -6.64607562172573, longitude: -55.546874999999999 }
      ]
    }
  ]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Repositioning the marker using drag and drop

The markers on the map can be dragged and dropped to change their position. To enable marker drag and drop, set the [enableDrag](#) property to **true** in the [markerSettings](#) property.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeSettings: {
      fill: '#C3E6ED'
    },
    markerSettings: [
      {
        enableDrag: true,
        dataSource: [
          { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'MarkerOne' },
          { latitude: 59.88893689676585, longitude: -109.3359375,
name: 'MarkerTwo' },
          { latitude: -6.64607562172573, longitude: -
55.54687499999999, name: 'MarkerThree' },
          { latitude: 23.644385824912135, longitude:
77.83189239539234, name: 'MarkerFour' },
          { latitude: 63.66569332894224, longitude:
98.2225173953924, name: 'MarkerFive' }
        ],
        visible: true,
        animationDuration: 0,
        shape: 'Balloon',
        width: 20,
        height: 20,
        border: { width: 2, color: '#285255' },
        tooltipSettings: {
          visible: true,
          valuePath: 'name',
        }
      },
    ],
  }],
});

```

```

    }}
  });
  map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

The data of the drag and dropped marker in the marker data source can be customized using the [markerDragStart](#) and [markerDragEnd](#) events. When you change the appropriate marker data, the tooltip and legend item text of that marker are automatically updated. The following properties are available in the event argument of the marker drag events.

Argument Name	Description
dataIndex	It represents the index of the data of the dragged marker in the marker data source.
latitude	It represents the latitude coordinate point of the dragged marker.
longitude	It represents the longitude coordinate point for the dragged marker.
markerIndex	It represents the index of the marker setting.

layerIndex	It represents the index of the layer in which the marker belongs.	
name	It represents the name of the event.	
x	It represents the horizontal location of the mouse pointer on the map when the drag action is performed.	
y	It represents the vertical location of the mouse pointer on the map when the drag action is performed.	

The following example shows how to use marker drag events to customize the data of the drag and dropped marker in the marker data source.

INDEX.JS

```
var map = new ej.maps.Maps({
  legendSettings:{
    visible:true,
    type:'Markers',
    shape:'Circle',
    shapeWidth:10,
    shapeHeight:10,
    fill:'#FF471A',
    shapeBorder: { width: 2, color: '#285255' },
  },
  markerDragStart: function(args){
    // When the marker begins to move on the map, the event is triggered.
  },
  markerDragEnd: function(args) {
    // When the marker on the map stops dragging, the event is triggered.
    var mapsInstance =
document.getElementById('element').ej2_instances[0];

mapsInstance.layers[args.layerIndex].markerSettings[args.markerIndex].dataSo
urce[args.dataIndex].name = 'Dragged Marker ' + (args.dataIndex + 1);
mapsInstance.refresh();
  },
  layers: [{
    shapeData: world_map,
    shapeSettings: {
      fill: '#C3E6ED'
    },
  },
  markerSettings: [
    {
      enableDrag: true,
      legendText: 'name',
      dataSource: [
        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'MarkerOne' },
        { latitude: 59.88893689676585, longitude: -109.3359375,
name: 'MarkerTwo'},
        { latitude: -6.64607562172573, longitude: -
55.546874999999999 , name: 'MarkerThree'},
        { latitude: 23.644385824912135, longitude:
77.83189239539234 , name: 'MarkerFour'},
        { latitude: 63.66569332894224, longitude:
98.2225173953924 , name: 'MarkerFive'}
```

```

        ],
        visible: true,
        animationDuration: 0,
        shape: 'Balloon',
        width: 20,
        height: 20,
        border: { width: 2, color: '#285255' },
        tooltipSettings: {
            visible: true,
            valuePath: 'name',
        }
    },
]
}]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Marker zooming

The Maps is initially scaled to the center value based on the marker distance. This can be achieved by setting the [shouldZoomInitially](#) property in [zoomSettings](#) as **true**.

INDEX.JS

```

var map = new ej.maps.Maps({

```

```

        zoomSettings: {
            enable: true,
            horizontalAlignment: 'Near',
            shouldZoomInitially: true
        },
        layers: [{
            shapeData: world_map,
            markerSettings: [{
                visible: true,
                dataSource: [
                    { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe' },
                    { latitude: 59.88893689676585, longitude: -109.3359375,
name: 'North America' },
                    { latitude: -6.64607562172573, longitude: -
55.546874999999999, name: 'South America' }
                ]
            }],
        }]
    });
    map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Marker clustering

Maps provide support to hide and cluster markers when they overlap each other. The number on a cluster indicates how many overlapped markers it contains. If zooming any of the cluster locations in Maps, the number on the cluster will decrease and begin to see the individual markers on the map. When zooming out, the overlapping marker will increase so that it can cluster again and increase the count over the cluster.

To enable clustering in markers, set the [allowClustering](#) property of [markerClusterSettings](#) as **true** and customization of clustering can be done with the [markerClusterSettings](#).

INDEX.JS

```
var maps = new ej.maps.Maps({
  useGroupingSeparator: true,
  format: 'n',
  zoomSettings: {
    enable: true
  },
  titleSettings: {
    text: 'Top 13 largest cities in the World',
    textStyle: {
      size: '16px'
    }
  },
  layers: [
    {
      shapeData: world_map,
      shapeSettings: {
        fill: '#C1DFF5'
      },
      markerClusterSettings: {
        allowClustering: true,
        shape: 'Circle',
        height: 40,
        width: 40,
        labelStyle : { color: 'white' },
      },
      markerSettings: [
        {
          visible: true,
          dataSource: cluster,
          shape: 'Balloon',
          tooltipSettings: {
            visible: true,
            valuePath: 'area',
            format: 'City: ${city} <br> Area: ${area}'
          },
          height: 20,
          width: 20,
          animationDuration: 0
        },
      ],
    }
  ],
});
maps.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="cluster.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Customization of marker cluster

The following properties are available to customize the marker clustering in the Maps component.

- [border](#) - To customize the color, width and opacity of the border of cluster in Maps.
- [connectorLineSettings](#) - To customize the connector line in cluster separating the markers.
- [dashArray](#) - To customize the dash array for the marker cluster in Maps.
- [fill](#) - Applies the color of the cluster in Maps.
- [height](#) - To customize the height of the marker cluster in Maps.
- [imageUrl](#) - To customize the URL path for the marker cluster when the cluster shape is set as image in Maps.
- [labelStyle](#) - To customize the text in marker cluster.
- [offset](#) - To customize the offset position for the marker cluster in Maps.
- [opacity](#) - To customize the opacity of the marker cluster.
- [shape](#) - To customize the shape for the cluster of markers.
- [width](#) - To customize the width of the marker cluster in Maps.

INDEX.JS

```

var maps = new ej.maps.Maps({

```

```

zoomSettings: {
  enable: true
},
layers: [{
  shapeData: world_map,
  shapeSettings: {
    fill: '#C1DFF5'
  },
  markerClusterSettings: {
    allowClustering: true,
    allowClusterExpand: true,
    shape: 'Circle',
    height: 40,
    width: 40,
    labelStyle : { color: 'white'},
    opacity: 0.9,
    fill: 'green',
    connectorLineSettings: {
      color: 'orange',
      opacity: 0.8,
      width: 2
    }
  },
  markerSettings: [{
    visible: true,
    dataSource: cluster,
    shape: 'Balloon',
    tooltipSettings: {
      format: 'City: ${city} <br> Area: ${area}',
      visible: true,
      valuePath: 'area',
    },
    height: 20,
    width: 20,
    animationDuration: 0
  }]
}]
});
maps.appendTo('#container');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="cluster.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Marker cluster expand

The cluster is formed by grouping an identical and non-identical marker from the surrounding area. By clicking on the cluster and setting the [allowClusterExpand](#) property in [markerClusterSettings](#) as **true** to expand the identical markers. If zoom in any of the locations of the cluster, the number on the cluster will decrease and the overlapping marker will be split into an individual marker on the map. When performing zoom out, it will increase the marker count and then cluster it again.

INDEX.JS

```

var map = new ej.maps.Maps({
  zoomSettings: {
    enable: true,
    mouseWheelZoom : true,
  },
  layers: [{
    shapeData: world_map,
    markerClusterSettings: {
      allowClustering: true,
      allowClusterExpand : true,
      shape: 'Circle',
      height: 40,
      width: 40,
      labelStyle : { color: 'white' },
    },
    markerSettings: [{
      visible: true,
      dataSource: [
        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe' },
        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe' },
        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe' },
        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe' },
        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe' },

```

```

        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe' },
        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe' },
        { latitude: 59.88893689676585, longitude: -109.3359375,
name: 'North America' },
        { latitude: -6.64607562172573, longitude: -
55.546874999999999, name: 'South America' }
    ]
  }],
  });
map.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Tooltip for marker

Tooltip is used to display more information about a marker on mouse over or touch end event. This can be enabled separately for marker by setting the [visible](#) property or [tooltipSettings](#) to **true**. The [valuePath](#) property in the [tooltipSettings](#) takes the field name that presents in dataSource and displays that value as tooltip text.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [
```



```

        {
            shapeData: usa_map,
            markerSettings: [
                {
                    dataSource: [
                        { latitude: 40.7424509, longitude: -74.0081468,
city: 'New York' }
                    ],
                    visible:true,
                    shape:'Circle',
                    fill:'white',
                    width:3,
                    animationDuration:0,
                    border: { width:2, color:'green'},
                    tooltipSettings: {
                        visible: true,
                        valuePath:'city'
                    }
                }
            ]
        },
        height: '450px',
        width: '700px'
    });
    map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>

```

```
</html>
```

See Also

- [Add different types of markers](#)
- [Tooltip for marker](#)

Bubble in EJ2 JavaScript Maps control

Bubbles in the Maps control represent the underlying data values of the Maps. It can be scattered throughout the Maps shapes that contain values in the data source. Bubbles are enabled by setting the [visible](#) property of [bubbleSettings](#) to **true**.

To add bubbles to the Maps, bind the data source to the [dataSource](#) property of [bubbleSettings](#) and set the field name, that contains the numerical data, in the data source to the [valuePath](#) property.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    bubbleSettings: [{
      visible: true,
      minRadius: 20,
      dataSource: [
        { name: 'India', population: '38332521' },
        { name: 'Burundi', population: '10524117' },
        { name: 'Pakistan', population: '3090416' }
      ],
      maxRadius: 40,
      valuePath: 'population'
    }]
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Bubble shapes

The following types of shapes are available to render the bubbles in Maps.

- Circle
- Square

By default, bubbles are rendered in the **Circle** type. To change the type of the bubble, set the [bubbleType](#) property of [bubbleSettings](#) as **Square** to render the square shape bubbles.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    bubbleSettings: [{
      bubbleType: 'Square',
      visible: true,
      dataSource: [
        { name: 'India', population: '38332521' },
        { name: 'Pakistan', population: '3090416' }
      ],
      valuePath: 'population'
    }]
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="world-map.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Customization

The following properties are available in [bubbleSettings](#) to customize the bubbles of the Maps component.

- [border](#) – To customize the color, width and opacity of the border of the bubbles in Maps.
- [fill](#) – To apply the color for bubbles in Maps.
- [opacity](#) – To apply opacity to the bubbles in Maps.
- [animationDelay](#) - To change the time delay in the transition for bubbles.
- [animationDuration](#) - To change the time duration of animation for bubbles.
- [highlightSettings](#) - To customize the highlight settings for the bubbles in Maps.
- [selectionSettings](#) - To customize the selection settings for the bubbles in Maps.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    bubbleSettings: [{
      visible: true,
      minRadius: 5,
      dataSource: [
        { name: 'India', population: '38332521' },
        { name: 'New Zealand', population: '19651127' },
        { name: 'Pakistan', population: '3090416' }
      ],
      fill: 'green',
      animationDelay: 100,
      animationDuration: 1000,
      maxRadius: 80,
      border: {
        color: 'blue',
        width: 2
      }
    }
  ]
});

```

```

        },
        opacity: 1,
        valuePath: 'population'
    }
  ]
}
});
map.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Setting color to the bubbles from the data source

The color for each bubble in the Maps can be set using the [colorValuePath](#) property of [bubbleSettings](#). The value for the [colorValuePath](#) property is the field name from the data source of the [bubbleSettings](#) which contains the color values.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    bubbleSettings: [{
      visible: true,
      minRadius: 20,
      colorValuePath: 'color',
```

```

        dataSource: [
            { name: 'India', population: '38332521', color: 'blue' },
            { name: 'New Zealand', population: '19651127', color:
'#c2d2d6' },
            { name: 'Pakistan', population: '3090416', color: '#09156d'
}
        ],
        maxRadius: 40,
        valuePath: 'population'
    }
}
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Bubble sizing

The size of the bubbles is calculated from the values got from the [valuePath](#) property. The range for the radius of the bubbles can be modified using [minRadius](#) and [maxRadius](#) properties.

INDEX.JS

```

var map = new ej.maps.Maps({
    layers: [{
        shapeData: world_map,
        shapeDataPath: 'name',
    }

```

```

        shapePropertyPath: 'name',
        bubbleSettings: [{
            visible: true,
            minRadius: 5,
            dataSource: [
                { name: 'India', population: '38332521' },
                { name: 'New Zealand', population: '19651127' },
                { name: 'Pakistan', population: '3090416' }
            ],
            maxRadius: 80,
            valuePath: 'population'
        }]
    });
    map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Multiple bubble groups

Multiple groups of bubbles can be added to the Maps using the [bubbleSettings](#) in which the properties of bubbles are added as an array. The customization for the bubbles can be done with the [bubbleSettings](#). In the following example, the gender-wise population ratio is demonstrated with two different bubble groups.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    bubbleSettings: [{
      visible: true,
      minRadius: 5,
      valuePath: "femaleRatio",
      colorValuePath: "femaleRatioColor",
      dataSource: [
        {
          country: "United States", femaleRatio: 50.50442726,
maleRatio: 49.49557274, femaleRatioColor: "green", maleRatioColor: "blue"
        },
        {
          country: "India", femaleRatio: 48.18032713, maleRatio:
51.81967287, femaleRatioColor: "blue", maleRatioColor: "#c2d2d6"
        },
        {
          country: "Oman", femaleRatio: 34.15597234, maleRatio:
65.84402766, femaleRatioColor: "#09156d", maleRatioColor: "orange"
        },
        {
          country: "United Arab Emirates", femaleRatio:
27.59638942, maleRatio: 72.40361058, femaleRatioColor: "#09156d",
maleRatioColor: "orange"
        }
      ],
      maxRadius: 20,
    },
    {
      visible: true,
      bubbleType: 'Circle',
      opacity: 0.4,
      minRadius: 15,
      valuePath: "maleRatio",
      colorValuePath: "maleRatioColor",
      dataSource: [
        {
          country: "United States", femaleRatio: 50.50442726,
maleRatio: 49.49557274, femaleRatioColor: "green", maleRatioColor: "blue"
        },
        {
          country: "India", femaleRatio: 48.18032713, maleRatio:
51.81967287, femaleRatioColor: "blue", maleRatioColor: "#c2d2d6"
        },
        {
          country: "Oman", femaleRatio: 34.15597234, maleRatio:
65.84402766, femaleRatioColor: "#09156d", maleRatioColor: "orange"
        },
        {
          country: "United Arab Emirates", femaleRatio:
27.59638942, maleRatio: 72.40361058, femaleRatioColor: "#09156d",
maleRatioColor: "orange"
        }
      ],
    }
  ]
});

```



```

        maxRadius: 25,
    }
  }
});
map.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Enable tooltip for bubble

The tooltip for the bubbles can be enabled by setting the [visible](#) property of the [tooltipSettings](#) as **true**. The content for the tooltip can be set using the [valuePath](#) property in the [tooltipSettings](#) of the [bubbleSettings](#) where the value for the [valuePath](#) property is the field name from the data source of the [bubbleSettings](#). Also added any HTML element as the template in tooltip using the [template](#) property.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    bubbleSettings: [{
      visible: true,
      minRadius: 5,
      dataSource: [
        { name: 'India', population: '38332521' },
```

```

        { name: 'New Zealand', population: '19651127' },
        { name: 'Pakistan', population: '3090416' }
    ],
    maxRadius: 80,
    valuePath: 'population',
    tooltipSettings: {
        visible: true,
        valuePath: 'population',
    }
    ]]
    ]]);
map.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Legend in EJ2 JavaScript Maps control

A Legend is a key on a Maps that contains descriptions for swatches of symbols. It can be represented in various colors, shapes or other identifiers based on the data and provides valuable information for interpreting what the Maps are displaying. It explains what each symbol in the Maps represents. Legends are enabled by setting the [visible](#) property of [legendSettings](#) to **true**.

Positioning of the legend

The legend can be positioned in the following two ways:

- Absolute position.
- Dock position.

Absolute position

The legend of the Maps can be positioned using the [location](#) property in the [legendSettings](#), which is based on the margin values of the X and Y axes. For positioning the legend based on margins corresponding to a Maps, the [position](#) property is set as **Float**.

Dock position

Legends are positioned in the following locations within the container. The [position](#) property in [legendSettings](#) is used to set these options in Maps.

- Top
- Left
- Bottom
- Right

The above four positions can be aligned combination of **Near**, **Center** and **Far** using [alignment](#) property in [legendSettings](#) property. So, the legend can be aligned to 12 positions.

INDEX.JS

```
var map = new ej.maps.Maps({
  titleSettings: {
    text: 'USA Election Results - 2016'
  },
  layers: [
    {
      shapeData: world_map,
      dataSource: [
        { "Country": "China", "Membership": "Permanent" },
        { "Country": "France", "Membership": "Permanent" },
        { "Country": "Russia", "Membership": "Permanent" },
        { "Country": "Kazakhstan", "Membership": "Non-Permanent" },
        { "Country": "Poland", "Membership": "Non-Permanent" },
        { "Country": "Sweden", "Membership": "Non-Permanent" } ],
      shapePropertyPath: 'name',
      shapeDataPath: 'Country',
      shapeSettings: {
        colorValuePath: 'Membership',
        colorMapping: [
          { value: 'Permanent', color: '#D84444' },
          { value: 'Non-Permanent', color: '#316DB5' }
        ]
      }
    }
  ],
  legendSettings: {
    visible: true,
    position: 'Top',
    alignment: 'Near'
  }
});
```

```
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Legend mode

Legend had two types of mode. **Default** mode and **Interactive** mode.

Default mode

Default mode legends having symbols with legend labels, used to identify the shape or bubble or marker color. To enable this option by setting the [mode](#) property of [legendSettings](#) as **Default**.

Interactive mode

The legends can be made interactive with an arrow mark indicating the exact range color in the legend when the mouse hovers over the corresponding shapes. To enable this type of mode by setting the [mode](#) property of [legendSettings](#) as **Interactive**. The [invertedPointer](#) property is used to enable or disable the visibility of the inverted pointer in interactive legend in Maps.

INDEX.JS

```
var map = new ej.maps.Maps({
  titleSettings: {
    text: 'USA Election Results - 2016'
  },
  layers: [
    {
```

```

        shapeData: world_map,
        dataSource: [
            { "Country": "China", "Membership": "Permanent" },
            { "Country": "France", "Membership": "Permanent" },
            { "Country": "Russia", "Membership": "Permanent" },
            { "Country": "Kazakhstan", "Membership": "Non-Permanent" },
            { "Country": "Poland", "Membership": "Non-Permanent" },
            { "Country": "Sweden", "Membership": "Non-Permanent" }],
        shapePropertyPath: 'name',
        shapeDataPath: 'Country',
        shapeSettings: {
            colorValuePath: 'Membership',
            colorMapping: [
                { value: 'Permanent', color: '#D84444' },
                { value: 'Non-Permanent', color: '#316DB5' }
            ]
        },
    ],
    legendSettings: {
        visible: true,
        mode: 'Interactive',
        invertedPointer: true
    }
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>

```

```
</html>
```

Legend for shapes

Legend for shapes can be generated from color mapping types such as equal color mapping, range color mapping and desaturation color mapping. Refer to the below data source which demonstrates the permanent and non-permanent countries in the UN security council.

```
`ts
```

```
export var unCountries: object[] = [
  { Country: 'China', Membership: 'Permanent' },
  { Country: 'France', Membership: 'Permanent' },
  { Country: 'Russia', Membership: 'Permanent' },
  { Country: 'United Kingdom', Membership: 'Permanent' },
  { Country: 'United States', Membership: 'Permanent' },
  { Country: 'Bolivia', Membership: 'Non-Permanent' },
  { Country: 'Eq. Guinea', Membership: 'Non-Permanent' },
  { Country: 'Ethiopia', Membership: 'Non-Permanent' },
  { Country: "Côte d'Ivoire", Membership: 'Permanent' },
  { Country: 'Kazakhstan', Membership: 'Non-Permanent' },
  { Country: 'Kuwait', Membership: 'Non-Permanent' },
  { Country: 'Netherlands', Membership: 'Non-Permanent' },
  { Country: 'Peru', Membership: 'Non-Permanent' },
  { Country: 'Poland', Membership: 'Non-Permanent' },
  { Country: 'Sweden', Membership: 'Non-Permanent' },
];
```

The below code snippet demonstrate the equal color mapping legends for the shapes. To bind the given data source to the [dataSource](#) property of [layerSettings](#). Set the value of [shapePropertyPath](#) to **name** and [shapeDataPath](#) to **Country**. To enable equal color mapping set the [colorMapping](#) as an array in [shapeSettings](#). Finally, set the [visible](#) property of [legendSettings](#) as **true**. The [label](#) property in [colorMapping](#) property is used to set the text name for legend in Maps.

INDEX.JS

```
var map = new ej.maps.Maps({
  titleSettings: {
    text: 'USA Election Results - 2016'
  },
  layers: [
    {
      shapeData: world_map,
      dataSource: [
```

```

        { "Country": "China", "Membership": "Permanent" },
        { "Country": "France", "Membership": "Permanent" },
        { "Country": "Russia", "Membership": "Permanent" },
        { "Country": "Kazakhstan", "Membership": "Non-Permanent" },
        { "Country": "Poland", "Membership": "Non-Permanent" },
        { "Country": "Sweden", "Membership": "Non-Permanent" }
    ],
    shapePropertyPath: 'name',
    shapeDataPath: 'Country',
    shapeSettings: {
        colorValuePath: 'Membership',
        colorMapping: [
            { value: 'Permanent', color: '#D84444' },
            { value: 'Non-Permanent', color: '#316DB5' }
        ]
    }
},
legendSettings: {
    visible: true
}
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Legend Shape

Maps supports the following types of legend shapes. The [shape](#) property in the [legendSettings](#) can be used to change the type of legend shapes.

- Circle
- Rectangle
- Triangle
- Diamond
- Cross
- Star
- HorizontalLine
- VerticalLine
- Pentagon
- InvertedTriangle

The shape of legends can be customized by using the [shapeWidth](#), [shapeHeight](#), [shapeBorder](#) and [shapePadding](#) properties.

Customization

The following properties are available in legend to customize the legend shape and legend text in Maps.

- [background](#) - To customize the background color of the Legend.
- [border](#) - To customize the color, width and opacity of the border for the Legend.
- [fill](#) - To apply the color for the Legend.
- [labelDisplayMode](#) - To customize the display mode for the Legend text.
- [labelPosition](#) - To customize the position of the Legend text.
- [orientation](#) - To customize the orientation of the Legend.
- [textStyle](#) - To customize the text style for Legend.
- [title](#) - To apply the title for the Legend .
- [titleStyle](#) - To customize the style of the title for the Legend.
- [height](#) - To customize the height of the Legend.
- [width](#) - To customize the width of the Legend.
- [opacity](#) - To apply the opacity to the Legend.

INDEX.JS

```
var map = new ej.maps.Maps({
  titleSettings: {
    text: 'USA Election Results - 2016'
  },
  layers: [
    {
      shapeData: world_map,
      dataSource: [{ "Country": "China", "Membership": "Permanent" },
        { "Country": "France", "Membership": "Permanent" },
        { "Country": "Russia", "Membership": "Permanent" },
        { "Country": "Kazakhstan", "Membership": "Non-Permanent" },
        { "Country": "Poland", "Membership": "Non-Permanent" },
        { "Country": "Sweden", "Membership": "Non-Permanent" } ],
      shapePropertyPath: 'name',
      shapeDataPath: 'Country',
    }
  ]
});
```



```

        shapeSettings: {
            colorValuePath: 'Membership',
            colorMapping: [
                {
                    value: 'Permanent', color: '#D84444'
                },
                {
                    value: 'Non-Permanent', color: '#316DB5'
                }
            ]
        }
    ],
    legendSettings: {
        visible: true,
        background: 'green',
        border: {
            color: 'blue',
            width: 2
        },
        fill: 'orange',
        labelPosition: 'Before',
        orientation: 'Vertical',
        textStyle: {
            size: '12px',
            color: 'red',
            fontStyle: 'italic'
        },
        title: {
            description: 'Legend title',
            text: 'Legend'
        },
        titleStyle: {
            size: '12px',
            color: '#d6e341',
            fontStyle: 'italic'
        }
    }
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Legend for items excluded from color mapping

The legend can be enabled for items excluded from the color mapping using the [color](#) property in [colorMapping](#) property. Refer to the **population_density** data which demonstrates the population density of some countries.

```
`ts
export var population_density = [
...
{
'code': 'AE',
'value': 90,
'name': 'United Arab Emirates',
'population': 8264070,
'density': 99
},
{
'code': 'GB',
'value': 257,
'name': 'United Kingdom',
'population': 62041708,
'density': 255
},
{
'code': 'US',
'value': 34,
'name': 'United States',
```

```

'population': 325020000,
'density': 33
}
...
];
`

```

The following example shows how to enable legend for items excluded from the color mapping. In the following example, color mapping is added for the ranges from 0 to 200. If there are any records in the data source that are outside of this range, the color mapping will not be applied. To apply the color for these excluded items, set the [color](#) property alone in the [colorMapping](#). To enable legend for these items, set the [visible](#) property of [legendSettings](#) to **true**.

INDEX.JS

```

var map = new ej.maps.Maps({
  legendSettings: {
    visible: true
  },
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    dataSource: Population_Density,
    shapeSettings: {
      colorValuePath: 'density',
      colorMapping: [
        {
          from: 0, to: 100, color: 'rgb(153,174,214)',
        },
        {
          from: 101, to: 200, color: 'rgb(115,143,199)',
        },
        {
          color: 'rgb(77,112,184)'
        },
      ],
    },
  ]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="world-map.js"></script>
<script src="data.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container" style="height: 500px; width: 700px">
  <div id="element"></div>
</div>
<script>
  var ele = document.getElementById('container');
  if (ele) {
    ele.style.visibility = "visible";
  }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Hide desired legend items

Use the [showLegend](#) property in the [colorMapping](#) to show or hide the desired legend items in Maps. If the [showLegend](#) property is set to **false**, the legend item will be hidden. otherwise, it will be visible.

INDEX.JS

```

var map = new ej.maps.Maps({
  legendSettings:{
    visible: true,
  },
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    dataSource: Population_Density,
    shapeSettings: {
      colorValuePath: 'density',
      colorMapping: [
        {
          from: 0, to: 100, color: 'rgb(153,174,214)', showLegend:
true
        },
        {
          from: 101, to: 200, color: 'rgb(115,143,199)',
showLegend: false
        },
        {
          color: 'rgb(77,112,184)', showLegend: false
        }
      ]
    }
  }]
});
map.appendTo('#element');

```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="data.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Hide legend items based on data source value

Depending on the values provided in the data source, the legend items will be hidden or visible. Bind the field name that contains the visibility state in the data source to the [showLegendPath](#) property of the [legendSettings](#) to achieve this.

INDEX.JS

```
var map = new ej.maps.Maps({
  legendSettings:{
    visible:true,
    showLegendPath:'visibility'
  },
  layers: [
    {
      shapeData: world_map,
      dataSource: default_data,
      shapePropertyPath: 'continent',
      shapeDataPath: 'continent',
      shapeSettings: {
        colorValuePath: 'color'
      }
    }
  ]
});
```

```
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="data.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Bind legend item text from data source

To show the legend text based on values provided in the data source, use the [valuePath](#) property in the [legendSettings](#).

INDEX.JS

```
var map = new ej.maps.Maps({
  legendSettings: {
    visible: true,
    valuePath: 'continent'
  },
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'continent',
    shapePropertyPath: 'continent',
    dataSource: default_data,
    shapeSettings: {
      colorValuePath: 'color'
    }
  }]
});
```

```
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="data.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Hide duplicate legend items

To hide the duplicate legend items in Maps, set the [removeDuplicateLegend](#) property to **true** in the [legendSettings](#).

INDEX.JS

```
var map = new ej.maps.Maps({
  legendSettings: {
    visible: true,
    valuePath: 'continent',
    removeDuplicateLegend: true
  },
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'continent',
    shapePropertyPath: 'continent',
    dataSource: default_data,
    shapeSettings: {
      colorValuePath: 'color'
    }
  }]
});
```

```
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="data.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Toggle option in legend

The toggle option has been provided for legend. If the legend can be toggled, the given color will be changed to the corresponding map shape item. To enable the toggle options in Legend, set the [enable](#) property of the [toggleLegendSettings](#) to **true**.

The following properties are available to customize the toggle option in legend.

- [applyShapeSettings](#) – To apply the [fill](#) property value to the shape of the Maps when it is set to **true**.
- [fill](#) – To apply the color to the shape of the Maps for which legend item is toggled.
- [opacity](#) – To customize the transparency for the shapes for which legend item is toggled.
- [border](#) – To customize the color, width and opacity of the border of the shapes in Maps.

INDEX.JS

```
var map = new ej.maps.Maps({
  legendSettings: {
    visible: true,
    toggleLegendSettings: {
```



```

        enable: true,
        applyShapeSettings: false,
        border: {
            color: 'green',
            width: 2
        }
    },
    layers: [{
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        dataSource: Population_Density,
        shapeSettings: {
            colorValuePath: 'density',
            colorMapping: [
                {
                    from: 0, to: 100, color: 'rgb(153,174,214)',
                },
                {
                    from: 101, to: 200, color: 'rgb(115,143,199)',
                },
                {
                    color: 'rgb(77,112,184)'
                },
            ]
        }
    }]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="data.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {

```

```

        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Enable legend for bubble

To enable the legend for the bubble by setting the [visible](#) property of [legendSettings](#) property as **true** and [type](#) property of [legendSettings](#) as **Bubbles**.

INDEX.JS

```

var map = new ej.maps.Maps({
  legendSettings: {
    visible: true,
    type: 'Bubbles'
  },
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    bubbleSettings: [{
      visible: true,
      minRadius: 20,
      dataSource: [
        { color: 'green', name: 'India', population: '38332521' },
        { color: 'purple', name: 'Burundi', population: '10524117' }
      ],
      { color: 'blue', name: 'Pakistan', population: '3090416' }
    ],
    maxRadius: 40,
    colorValuePath: 'color',
    valuePath: 'population'
  }]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Enable legend for markers

To enable legend for marker by setting the [visible](#) property of [legendSettings](#) as **true** and [type](#) property of [legendSettings](#) as **Markers**. The [legendText](#) property in the [markerSettings](#) property can be used to show the legend text based on values provided in the data source.

INDEX.JS

```
var map = new ej.maps.Maps({
  legendSettings: {
    visible: true,
    type: 'Markers'
  },
  layers: [
    {
      shapeData: world_map,
      markerSettings: [
        {
          visible: true,
          legendText: 'name',
          dataSource: [
            { latitude: 37.6276571, longitude: -122.4276688,
name: 'San Bruno' },
            { latitude: 33.5302186, longitude: -117.7418381,
name: 'Laguna Niguel' },
            { latitude: 40.7424509, longitude: -74.0081468,
name: 'New York' }
          ],
          shape: 'Circle'
        }
      ]
    }
  ]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```

<title>EJ2 Maps</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<script src="world-map.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Imitate/Map marker shape to the legend shape

To imitate or map the marker shape with its legend item shape, set the [useMarkerShape](#) property to **true** in the [legendSettings](#) property.

INDEX.JS

```

var map = new ej.maps.Maps({
  legendSettings: {
    visible: true,
    type: 'Markers',
    useMarkerShape: true,
    toggleLegendSettings: {
      enable: true,
      applyShapeSettings: false,
      border: {
        color: 'green',
        width: 2
      }
    }
  },
  layers: [
    {
      shapeDataPath: 'name',
      shapePropertyPath: 'name',
      shapeData: world_map,
      shapeSettings: {
        fill: '#E5E5E5'
      },
      markerSettings: [
        {

```

```

        dataSource: markerDataSource,
        colorValuePath: 'color',
        shapeValuePath: 'shape',
        legendText: 'country',
        visible: true
    }
]
}
]
});
map.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="markerdata.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Annotations in EJ2 JavaScript Maps control

<!-- markdownlint-disable MD013 -->

Annotations are used to mark the specific area of interest in the Maps with texts, shapes, or images. Any number of annotations can be added to the Maps component.

Annotation

By using the [content](#) property of [annotation](#), text content or id of an element or an HTML string can be specified to render a new HTML element in Maps.

<!-- markdownlint-disable MD036 -->

```

<script id='annotation' type='text/x-template'>
<div id='template'>
<img src='src/maps/images/flight.png'>
</div>
</script>

```

```

`ts
var maps = new ej.maps.Maps({
  annotations: [
    {
      content: '#annotation',
      x: '0%', y: '50%'
    }
  ],
  layers: [
    {
      shapeData: world_map,
    }
  ]
});
maps.appendTo('#element');

```

Annotation customization

Changing the z-index

The stack order of an annotation element can be changed using the [zIndex](#) property in the [annotation](#).

INDEX.JS

```

var map = new ej.maps.Maps({
  annotations: [{
    content: '<div id="first"><h1>Maps</h1></div>',
    x: '0%', y: '50%',
    zIndex: '-1'
  }],
  layers: [{
    shapeData: world_map
  }]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

<!-- markdownlint-disable MD036 -->

Positioning an Annotation

Annotations can be placed anywhere in the Maps by specifying pixel or percentage values to the [x](#) and [y](#) properties in the [annotation](#).

INDEX.JS

```

var map = new ej.maps.Maps({
  annotations: [{
    content: '<div id="first"><h1>Maps</h1></div>',
    x: '20%', y: '50%',
    zIndex: '-1'
  }],
  layers: [{
    shapeData: world_map
  }]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>

```

```

<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

<!-- markdownlint-disable MD036 -->

Alignment of an Annotation

Annotations can be aligned using the [horizontalAlignment](#) and [verticalAlignment](#) properties in the [annotation](#). The possible values can be **Center**, **Far**, **Near** and **None**.

INDEX.JS

```

var map = new ej.maps.Maps({
  annotations: [{
    content: '<div id="first"><h1>Maps</h1></div>',
    verticalAlignment: 'Center',
    horizontalAlignment: 'Center',
    x: '20%', y: '50%',
    zIndex: '-1'
  }],
  layers: [{
    shapeData: world_map
  }]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>

```



```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<script src="world-map.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Multiple Annotation

Multiple annotations can be added to the Maps using the [annotations](#) and customization for the annotations can be done with this property.

INDEX.JS

```

var map = new ej.maps.Maps({
  annotations: [{
    content: '<div id="first"><h1>Maps-Annotation</h1></div>',
    x: '50%', y: '0%',
    zIndex: '-1'
  },
  {
    content: '<div id="first"><h1>Maps</h1></div>',
    x: '20%', y: '50%',
    zIndex: '-1'
  }],
  layers: [{
    shapeData: world_map
  }]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<script src="world-map.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Navigation line in EJ2 JavaScript Maps control

The navigation lines are used to denote the path between two locations. This feature can be used to draw flight or sea routes. Navigation lines are enabled by setting the [visible](#) property of the [navigationLineSettings](#) to **true**.

Customization

The following properties are available in [navigationLineSettings](#) to customize the navigation line of the Maps component.

- [color](#) - To apply the color for navigation lines in Maps.
- [dashArray](#) - To define the pattern of dashes and gaps that is applied to the outline of the navigation lines.
- [width](#) - To customize the width of the navigation lines.
- [angle](#) - To customize the angle of the navigation lines.
- [highlightSettings](#) - To customize the highlight settings of the navigation line.
- [selectionSettings](#) - To customize the selection settings of the navigation line.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [{
    navigationLineSettings: [{
      visible: true,
      latitude: [40.7128, 36.7783],
      longitude: [-74.0060, -119.4179],
      color: 'black',
      angle: 90,
      width: 2,
      dashArray: '4'
    }
  ]
});

```

```

    }],
    shapeData: world_map,
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="usa.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Enabling the arrows

To enable the arrow in the navigation line, set the [showArrow](#) property of [arrowSettings](#) to **true**. The following properties are available in [arrowSettings](#) to customize the arrow of the navigation lines.

- [color](#) - To apply the color for arrow of the navigation line.
- [offset](#) - To customize the offset position of the arrow of the navigation line.
- [position](#) - To customize the position of the arrow in navigation line. The possible values can be **Start** and **End**.
- [size](#) - To customize the size of the arrow in pixels.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [{
    navigationLineSettings: [{
      visible: true,
      latitude: [40.7128, 36.7783],
```

```

        longitude: [-74.0060, -119.4179],
        color: 'black',
        angle: 90,
        width: 2,
        dashArray: '4',
        arrowSettings: {
            showArrow: true,
            size: 15,
            position: 'Start'
        }
    }],
    shapeData: world_map,
}]]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

User interactions in EJ2 JavaScript Maps control

Zooming, panning, single and double click, highlight and selection are all options that allow for effective interaction with Map elements.

Zooming

The zooming feature is used to zoom in and out of Maps to show in-depth information. It is controlled by the [zoomFactor](#) property of the [zoomSettings](#) of the Maps. The [zoomFactor](#) is increased or decrease dynamically based on zoom in and out interaction.

Enable Zooming

Zooming of Maps is enabled by setting the [enable](#) property of [zoomSettings](#) to **true**. To enable Zooming in Maps, the **Zoom** module must be injected into Maps using **Maps.Inject(Zoom)** method.

<!-- markdownlint-disable MD010 -->

Enable panning

To enable the panning feature, set the [enablePanning](#) property of [zoomSettings](#) to **true**.

```
`ts
var map = new ej.maps.Maps({
  zoomSettings: {
    enable: true,
    enablePanning:true
  },
  layers: [{
    shapeData: world_map,
  }]
});
map.appendTo('#element');
```

Various type of zooming

Zooming can be performed in following types:

Zooming toolbar

By default, the toolbar is rendered with **zoom-in**, **zoom-out**, and **reset** options when it is set to **true** in the [enable](#) property of [zoomSettings](#).

The following options are available in toolbar.

1. Zoom - Provides rectangular zoom support.
2. ZoomIn - Zooms in the Maps.
3. ZoomOut - Zooms out the Maps.
4. Pan - Switches to panning if rectangular zoom is activated.
5. Reset - Restores the Maps to the default view.

Refer the [API](#) links for Zooming.

```
`ts
var map = new ej.maps.Maps({
```

```
zoomSettings: {  
  enable: true,  
  buttonSettings: {  
    toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset']  
  }  
},  
layers: [{  
  shapeData: usa_map,  
}]  
});  
map.appendTo('#element');
```

Pinch zooming

To enable or disable the pinch zooming, use the [pinchZooming](#) property in [zoomSettings](#).

```
`ts
```

```
var map = new ej.maps.Maps({  
  zoomSettings: {  
    enable: true,  
    pinchZooming: true  
  },  
  layers: [{  
    shapeData: usa_map,  
  }]  
});  
map.appendTo('#element');
```

Single-click zooming

To enable or disable the single-click zooming, use the [zoomOnClick](#) property in [zoomSettings](#).

```
`ts
```

```
var map = new ej.maps.Maps({  
  zoomSettings: {  
    enable: true,  
    zoomOnClick: true  
  }  
});  
map.appendTo('#element');
```

```
},  
layers: [{  
  shapeData: usa_map,  
}]  
});  
map.appendTo('#element');  
`
```

Double-click zooming

To enable or disable the double-click zooming, use the [doubleClickZoom](#) property in [zoomSettings](#).

```
`ts  
var map = new ej.maps.Maps({  
  zoomSettings: {  
    enable: true,  
    doubleClickZoom: true  
  },  
  layers: [{  
    shapeData: usa_map,  
  }]  
});  
map.appendTo('#element');  
`
```

Mouse wheel zooming

To enable or disable mouse wheel zooming, use the [mouseWheelZoom](#) property in [zoomSettings](#).

```
`ts  
var map = new ej.maps.Maps({  
  zoomSettings: {  
    enable: true,  
    mouseWheelZoom: true  
  },  
  layers: [{  
    shapeData: usa_map,  
  }]  
});
```

```
map.appendTo('#element');  
`ts
```

Minimum and maximum zooming

The zooming range can be adjusted using the [minZoom](#) and [maxZoom](#) properties in [zoomSettings](#). The minZoom value is set to **1** by default, and the maxZoom value is set to **10**.

```
`ts  
var map = new ej.maps.Maps({  
  zoomSettings: {  
    enable: true,  
    minZoom: 2,  
    maxZoom: 12  
  },  
  layers: [{  
    shapeData: usa_map,  
  }]  
});  
map.appendTo('#element');  
`ts
```

Zooming with animation

To zoom in or zoom out the Maps with animation, use the [animationDuration](#) property in [layers](#).

```
`ts  
var map = new ej.maps.Maps({  
  zoomSettings: {  
    enable: true,  
  },  
  layers: [{  
    shapeData: usa_map,  
    animationDuration: 500  
  }]  
});  
map.appendTo('#element');  
`ts
```

Customizing the zoom toolbar

The zoom toolbar can be customized by using the [toolbarSettings](#) option in the [zoomSettings](#). The following properties can be used to customize the zoom toolbar.

- [backgroundColor](#) - It is used to customize the background color of the zoom toolbar.
- [borderOpacity](#) - It is used to customize the opacity of the border of the zoom toolbar.
- [borderWidth](#) - It is used to customize the thickness of the border of the zoom toolbar.
- [borderColor](#) - It is used to customize the color of the border of the zoom toolbar.
- [horizontalAlignment](#) - It is used to position the zoom toolbar in near, far, and center positions to customize its horizontal placement.
- [verticalAlignment](#) - It is used to position the zoom toolbar in near, far, and center positions to customize its vertical placement.
- [orientation](#) - It is used to change the orientation (horizontal/vertical) of the zoom toolbar.

INDEX.JS

```
var map = new ej.maps.Maps({
  zoomSettings: {
    enable: true,
    toolbarSettings: {
      orientation: 'Vertical',
      backgroundColor: 'pink',
      borderWidth: 3,
      borderColor: 'green',
      verticalAlignment: 'Near',
      buttonSettings: {
        toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset']
      }
    }
  },
  layers: [
    {
      shapeData: world_map,
      shapeSettings: {
        fill: '#C1DFF5'
      }
    }
  ]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

<!-- markdownlint-disable MD036 -->

Customizing the buttons in the zoom toolbar

The appearance of the buttons in the zoom toolbar can be customized by using the [buttonSettings](#) option in the [toolbarSettings](#) of the [zoomSettings](#) property. The following properties can be used to customize the zoom toolbar buttons.

- [fill](#) - It is used to set the background color of the buttons.
- [color](#) - It is used to customize the color of the icons inside the button.
- [borderOpacity](#) - It is used to set the opacity of the border of the zoom toolbar buttons.
- [borderWidth](#) - It is used to set the thickness of the border of the zoom toolbar buttons.
- [borderColor](#) - It is used to set the color of the border of the zoom toolbar buttons.
- [radius](#) - It is used to set the size of the button.
- [selectionColor](#) - It is used to set the color of the icons inside the button when selection is performed.
- [highlightColor](#) - It is used to change the color of the button when the mouse is hovered over it.
- [padding](#) - It is used to change the padding space between each button.
- [opacity](#) - It is used to change the opacity of the button.
- [toolbarItems](#) - It is used to change the items that should be displayed in the zoom toolbar. By default, zoom-in, zoom-out, and reset buttons will be available. Other options include selection zoom and panning.

INDEX.JS

```
var map = new ej.maps.Maps({
  zoomSettings: {
    enable: true,
    toolbarSettings: {
      buttonSettings: {
        fill: 'pink',
        padding: 10,
        toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset'],
        color: 'red',
        borderColor: 'green',
        radius: 35,
        selectionColor: '#d55e5e',
        highlightColor: '#5ed59a',
        opacity: 0.6,
      }
    }
  }
});
```

```

        borderWidth: 2
    }
}
},
layers: [
    {
        shapeData: world_map,
        shapeSettings: {
            fill: '#C1DFF5'
        }
    }
]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

<!-- markdownlint-disable MD036 -->

Customizing the tooltip of the zoom toolbar

The appearance of the tooltip of the zoom toolbar can be customized by using the [tooltipSettings](#) option in the [toolbarSettings](#) of the [zoomSettings](#) property. The following properties are available to customize the zoom toolbar tooltip.

- [visible](#) - Enables or disables the tooltip of the zoom toolbar.

- [fill](#) - It is used to change the background color of the tooltip of the zoom toolbar.
- [borderOpacity](#) - It is used to change the opacity of the border of the zoom toolbar's tooltip.
- [borderWidth](#) - It is used to change the thickness of the border of the zoom toolbar's tooltip.
- [borderColor](#) - It is used to change the color of the border of the zoom toolbar's tooltip.
- [fontColor](#) - It is used to change the color of the text in the tooltip of the zoom toolbar.
- [fontFamily](#) - It is used to change the font family of the text in the tooltip of the zoom toolbar.
- [fontStyle](#) - It is used to change the font style of the text in the tooltip of the zoom toolbar.
- [fontWeight](#) - It is used to change the font weight of the text in the tooltip of the zoom toolbar.
- [fontSize](#) - It is used to change the size of the text in the tooltip of the zoom toolbar.
- [fontOpacity](#) - It is used to change the opacity of the text in the tooltip of the zoom toolbar.

INDEX.JS

```
var map = new ej.maps.Maps({
  zoomSettings: {
    enable: true,
    toolbarSettings: {
      tooltipSettings: {
        visible: true,
        borderWidth: 2,
        borderColor: 'green',
        fontColor: 'black',
        fill: 'violet',
        fontFamily: 'Times New Roman',
        fontWeight: 200,
        fontSize: '22px',
        fontOpacity: 1
      },
      buttonSettings: {
        toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset']
      }
    }
  },
  layers: [
    {
      shapeData: world_map,
      shapeSettings: {
        fill: '#C1DFF5'
      }
    }
  ]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
<script src="world-map.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Selection

Each shape in the Maps can be selected and deselected during interaction with the shapes. Selection is enabled by setting the [enable](#) property of [selectionSettings](#) to **true**.

The following properties are available to customize the selection of Maps elements such as shapes, bubbles, markers and legends.

- [border](#) - To customize the color, width and opacity of the border of which element is selected in Maps.
- [fill](#) - Applies the color for the element that is selected.
- [opacity](#) - To customize the transparency for the element that is selected.
- [enableMultiSelect](#) - To enable or disable the selection for multiple shapes or markers or bubbles in the Maps.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [
    {
      shapeData: world_map,
      dataSource: [{ "Country": "China", "Membership": "Permanent" },
        { "Country": "France", "Membership": "Permanent" },
        { "Country": "Russia", "Membership": "Permanent" },
        { "Country": "Kazakhstan", "Membership": "Non-Permanent" },
        { "Country": "Poland", "Membership": "Non-Permanent" },
        { "Country": "Sweden", "Membership": "Non-Permanent" } ],
      shapePropertyPath: 'name',
      shapeDataPath: 'Country',
      selectionSettings: {
        enable: true,
        fill: 'blue',
        border: { color: 'white', width: 2 }
      }
    },
  ],
});

```

```

        shapeSettings: {
            colorValuePath: 'Membership',
            colorMapping: [
                {
                    value: 'Permanent', color: '#D84444'
                },
                {
                    value: 'Non-Permanent', color: '#316DB5'
                }
            ]
        },
    ],
    legendSettings: {
        visible: true
    }
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Enable selection for bubbles

To enable the selection for bubbles in Maps, set the [selectionSettings](#) in [bubbleSettings](#) and set the [enable](#) property of [selectionSettings](#) as **true**.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    bubbleSettings: [{
      visible: true,
      dataSource: [
        { name: 'India', population: '38332521' },
        { name: 'South Africa', population: '19651127' },
        { name: 'Pakistan', population: '3090416' }
      ],
      selectionSettings: {
        enable: true,
        fill: 'green',
        border: { color: 'white', width: 2 }
      },
      valuePath: 'population'
    }]
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Enable selection for markers

To enable the selection for markers in Maps, set the [selectionSettings](#) in the [markerSettings](#) and set the [enable](#) property of the [selectionSettings](#) as **true**.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    markerSettings: [{
      visible: true,
      height: 20,
      width: 20,
      fill: 'green',
      shape: 'Balloon',
      selectionSettings: {
        enable: true,
        fill: 'blue',
        border: { color: 'white', width: 2}
      },
      dataSource: [
        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe' },
        { latitude: 59.88893689676585, longitude: -109.3359375,
name: 'North America' },
        { latitude: -6.64607562172573, longitude: -
55.54687499999999, name: 'South America' }
      ]
    }
  ]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
```



```

        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Enable selection for polygons

When the [enable](#) property of [selectionSettings](#) is set to **true**, the polygon shapes can be selected via user interaction. The following properties are available in [selectionSettings](#) to customize the polygon shape when it is selected.

- [enableMultiSelect](#) - It is used to enable multiple selection of polygon shapes.
- [fill](#) - It is used to change the color of the selected polygon shape.
- [opacity](#) - It is used to change the opacity of the selected polygon shape.
- [border](#) - This property is used to change the color, width, and opacity of the border of the selected polygon shape.

The following example shows how to select the polygon shape in the geometry map.

INDEX.JS

```

var map = new ej.maps.Maps({
    layers: [
        {
            shapeData: world_map,
            polygonSettings: {
                polygons: [
                    {
                        points: [
                            { longitude: -1.8920678947185365, latitude:
35.06195799239681 },
                            { longitude: -1.6479633699113947, latitude:
33.58989612266137 },
                            { longitude: -1.4201220366858252, latitude:
32.819439646045254 },
                            { longitude: -1.197974596225663, latitude:
32.26940895444655 },
                            { longitude: -2.891112397949655, latitude:
32.10303058820031 },
                            { longitude: -3.8246984550501963, latitude:
31.34551662687602 },
                            { longitude: -3.720166273688733, latitude:
30.758086682848685 },
                            { longitude: -5.6571886081189575, latitude:
29.613582597203006 },
                            { longitude: -7.423353242214745, latitude:
29.44328441403087 },
                            { longitude: -8.6048931685323, latitude:
28.761444633616776 },
                            { longitude: -8.695726975465703, latitude:
27.353491085576195 },

```

```

19.15916564839422 }, { longitude: 3.837867279970908, latitude:
19.48749097192868 }, { longitude: 6.0705408799045415, latitude:
23.694596786078293 }, { longitude: 12.055736352807713, latitude:
24.289329186946034 }, { longitude: 11.272522332402986, latitude:
24.65419958524693 }, { longitude: 10.30872578261932, latitude:
25.48943950947175 }, { longitude: 9.910236690050027, latitude:
26.398372489836902 }, { longitude: 9.432639882414293, latitude:
26.73489453809293 }, { longitude: 9.898266456582292, latitude:
30.31040379467153 }, { longitude: 9.560243026853641, latitude:
32.350324876652195 }, { longitude: 8.943853847283322, latitude:
33.75071049019398 }, { longitude: 7.57004059025715, latitude:
34.69043151009983 }, { longitude: 8.0906322609153, latitude:
35.38654406371319 }, { longitude: 8.363285449347273, latitude:
36.44751078733985 }, { longitude: 8.26139549449448, latitude:
36.881913362940196 }, { longitude: 8.61100824823302, latitude:
37.021408008916254 }, { longitude: 7.4216488925819135, latitude:
36.99092409199429 }, { longitude: 6.461182254165351, latitude:
36.69985479014656 }, { longitude: 5.297178918070159, latitude:
36.86470546831693 }, { longitude: 3.6718056161224695, latitude:
36.57658056301722 }, { longitude: 1.2050052555659931, latitude:
35.806903541813625 }, { longitude: -0.26968570003779746, latitude:
35.58466127904214 }, { longitude: -0.995191786435754, latitude:
35.06195799239681 }, { longitude: -1.8920678947185365, latitude:
    ],
    fill: 'blue',
    opacity: 0.7,
    borderColor: 'green',
    borderWidth: 2,
    borderOpacity: 0.7
  }
],
highlightSettings: {
  enable: true,
  fill: 'blue',

```

```

        opacity: 0.7,
        border: {
            color: 'green',
            width: 2,
            opacity: 0.7
        }
    },
    selectionSettings: {
        enable: true,
        fill: 'violet',
        enableMultiSelect: false,
        opacity: 0.8,
        border: {
            color: 'cyan',
            opacity: 1,
            width: 7
        }
    }
}
}
]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Public method for the shape selection

The [shapeSelection](#) method can be used to select each shape in the Maps.

LayerIndex, propertyName, country name, and selected value as a boolean state(true / false) are the input parameters for this method.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    selectionSettings: {
      enable: true,
      fill: 'green',
      border: { color: 'white', width: 2}
    }
  }]
});
map.appendTo('#element');
document.getElementById('selection').onclick = () => {
  map.shapeSelection(0, "continent", "Asia", true);
};
document.getElementById('unselection').onclick = () => {
  map.shapeSelection(0, "continent", "Asia", false);
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
    <button id="selection" type="button" width="15%">Select</button>
    <button id="unselection" type="button"
width="15%">UnSelect</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Initial shape selection

The shape is initially selected using the [initialShapeSelection](#) property, and the values are mapped to the [shapePath](#) and [shapeValue](#).

`initialShapeSelection` is an Array property.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    initialShapeSelection: [
      { shapePath: 'continent', shapeValue: 'Africa' },
      { shapePath: 'name', shapeValue: 'India' }
    ],
    selectionSettings: {
      enable: true,
      fill: 'green',
      border: { color: 'white', width: 2 }
    }
  ]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
```

```
</html>
```

Initial marker selection

Using the [initialMarkerSelection](#) property, the marker shape can be selected initially. Markers render based on the [latitude](#) and [longitude](#) values.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    markerSettings: [{
      visible: true,
      height: 20,
      width: 20,
      fill: 'green',
      shape: 'Balloon',
      initialMarkerSelection: [{
        latitude: -6.64607562172573, longitude: -55.54687499999999
      }],
      selectionSettings: {
        enable: true,
        fill: 'blue',
        opacity: 1,
        border: { color: 'white', width: 2, opacity: 1 }
      },
      dataSource: [
        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe' },
        { latitude: 59.88893689676585, longitude: -109.3359375,
name: 'North America'},
        { latitude: -6.64607562172573, longitude: -
55.54687499999999, name: 'South America' }
      ]
    }],
  }]);
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Highlight

Each shape in the Map can be highlighted during mouse hover on the Map elements such as shapes, bubbles, markers and legends. Highlight is enabled by setting the [enable](#) property of [highlightSettings](#) to **true**.

The following properties are available to customize the highlight of Map elements such as shapes, bubbles, markers and legends.

- [border](#) - To customize the color, width and opacity of the border of which element is highlighted in Maps.
- [fill](#) - Applies the color for the element that is highlighted.
- [opacity](#) - To customize the transparency for the element that is highlighted.

Hovering on the specific legend, the shapes which are bounded to the selected legend is also highlighted and vice versa.

INDEX.JS

```

var map = new ej.maps.Maps({
  layers: [
    {
      shapeData: world_map,
      dataSource: [
        { "Country": "China", "Membership": "Permanent" },
        { "Country": "France", "Membership": "Permanent" },
        { "Country": "Russia", "Membership": "Permanent" },
        { "Country": "Kazakhstan", "Membership": "Non-Permanent" },
        { "Country": "Poland", "Membership": "Non-Permanent" },
        { "Country": "Sweden", "Membership": "Non-Permanent" }
      ],
      shapePropertyPath: 'name',
      shapeDataPath: 'Country',
      highlightSettings: {
        enable: true,
        fill: 'blue',
        border: { color: 'white', width: 2 }
      },
      shapeSettings: {
        colorValuePath: 'Membership',

```

```

        colorMapping: [
            {
                value: 'Permanent', color: '#D84444'
            },
            {
                value: 'Non-Permanent', color: '#316DB5'
            }
        ]
    },
    legendSettings: {
        visible: true
    }
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Enable highlight for bubbles

To enable the highlight for bubbles in Maps, set the [highlightSettings](#) in [bubbleSettings](#) and set the [enable](#) property of [highlightSettings](#) as **true**.

INDEX.JS

```

var map = new ej.maps.Maps({
    layers: [{

```



```

    shapeData: world_map,
    shapeDataPath: 'name',
    shapePropertyPath: 'name',
    bubbleSettings: [{
      visible: true,
      dataSource: [
        { name: 'India', population: '38332521' },
        { name: 'South Africa', population: '19651127' },
        { name: 'Pakistan', population: '3090416' }
      ],
      highlightSettings: {
        enable: true,
        fill: 'green',
        border: { color: 'white', width: 2 }
      },
      valuePath: 'population'
    }]
  });
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Enable highlight for markers

To enable the highlight for markers in Maps, set the [highlightSettings](#) in [markerSettings](#) and set the [enable](#) property of [highlightSettings](#) as **true**.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    markerSettings: [{
      visible: true,
      height: 20,
      width: 20,
      fill: 'green',
      shape: 'Balloon',
      highlightSettings: {
        enable: true,
        fill: 'blue',
        border: { color: 'white', width: 2}
      },
      dataSource: [
        { latitude: 49.95121990866204, longitude:
18.468749999999998, name: 'Europe' },
        { latitude: 59.88893689676585, longitude: -109.3359375,
name: 'North America'},
        { latitude: -6.64607562172573, longitude: -
55.54687499999999, name: 'South America'}
      ]
    }],
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Enable highlight for polygons

The polygon shapes can be highlighted via user interaction if the [enable](#) property of [highlightSettings](#) is set to **true**. The following properties are available in [highlightSettings](#) to customize the polygon shape when it is highlighted.

- [fill](#) - It is used to change the color of the highlighted polygon shape.
- [opacity](#) - It is used to change the opacity of the highlighted polygon shape.
- [border](#) - This property is used to change the color, width, and opacity of the border of the highlighted polygon shape.

The following example shows how to highlight a polygon shape on a geometry map.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [
    {
      shapeData: world_map,
      polygonSettings: {
        polygons: [
          {
            points: [
              { longitude: -1.8920678947185365, latitude:
35.06195799239681 },
              { longitude: -1.6479633699113947, latitude:
33.58989612266137 },
              { longitude: -1.4201220366858252, latitude:
32.819439646045254 },
              { longitude: -1.197974596225663, latitude:
32.26940895444655 },
              { longitude: -2.891112397949655, latitude:
32.10303058820031 },
              { longitude: -3.8246984550501963, latitude:
31.34551662687602 },
              { longitude: -3.720166273688733, latitude:
30.758086682848685 },
              { longitude: -5.6571886081189575, latitude:
29.613582597203006 },
              { longitude: -7.423353242214745, latitude:
29.443284411403087 },
              { longitude: -8.6048931685323, latitude:
28.761444633616776 },
              { longitude: -8.695726975465703, latitude:
27.353491085576195 },
              { longitude: 3.837867279970908, latitude:
19.15916564839422 },
              { longitude: 6.0705408799045415, latitude:
19.48749097192868 },
              { longitude: 12.055736352807713, latitude:
23.694596786078293 },
```

```

    { longitude: 11.272522332402986, latitude:
24.289329186946034 },
    { longitude: 10.30872578261932, latitude:
24.65419958524693 },
    { longitude: 9.910236690050027, latitude:
25.48943950947175 },
    { longitude: 9.432639882414293, latitude:
26.398372489836902 },
    { longitude: 9.898266456582292, latitude:
26.73489453809293 },
    { longitude: 9.560243026853641, latitude:
30.31040379467153 },
    { longitude: 8.943853847283322, latitude:
32.350324876652195 },
    { longitude: 7.57004059025715, latitude:
33.75071049019398 },
    { longitude: 8.0906322609153, latitude:
34.69043151009983 },
    { longitude: 8.363285449347273, latitude:
35.38654406371319 },
    { longitude: 8.26139549449448, latitude:
36.44751078733985 },
    { longitude: 8.61100824823302, latitude:
36.881913362940196 },
    { longitude: 7.4216488925819135, latitude:
37.021408008916254 },
    { longitude: 6.461182254165351, latitude:
36.99092409199429 },
    { longitude: 5.297178918070159, latitude:
36.69985479014656 },
    { longitude: 3.6718056161224695, latitude:
36.86470546831693 },
    { longitude: 1.2050052555659931, latitude:
36.57658056301722 },
    { longitude: -0.26968570003779746, latitude:
35.806903541813625 },
    { longitude: -0.995191786435754, latitude:
35.58466127904214 },
    { longitude: -1.8920678947185365, latitude:
35.06195799239681 }
    ],
    fill: 'red',
    opacity: 0.7,
    borderColor: 'green',
    borderWidth: 2,
    borderOpacity: 0.7
  }
],
highlightSettings: {
  enable: true,
  fill: 'yellow',
  opacity: 0.4,
  border: {
    color: 'blue',
    opacity: 0.6,
    width: 4
  }
}

```

```

        },
        selectionSettings: {
            enable: true,
            fill: 'red',
            opacity: 0.7,
            border: {
                color: 'green',
                width: 2,
                opacity: 0.7
            }
        }
    }
}
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Tooltip

On mouse over or touch end event, the tooltip is used to get more information about the layer, bubble, or marker. It can be enabled separately for layer or bubble or marker by using the [visible](#) property of [tooltipSettings](#) as **true**. The [valuePath](#) property in the tooltip takes the field name that presents in data source and displays that value as tooltip text. The [tooltipDisplayMode](#) property is used to change the

display mode of the tooltip in Maps. Following display modes of tooltip are available in the Maps component. By default, [tooltipDisplayMode](#) is set to **MouseMove**.

INDEX.JS

```
var map = new ej.maps.Maps({
  layers: [{
    shapeData: world_map,
    tooltipSettings: {
      visible: true,
      valuePath: 'name'
    }
  }]
});
map.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Customization

The following properties are available in the [tooltipSettings](#) to customize the tooltip of the Maps component.

- [border](#) - To customize the color, width and opacity of the border of the tooltip in layers, markers, and bubbles of Maps.
- [fill](#) - Applies the color of the tooltip in layers, markers, and bubbles of Maps.

- [format](#) - To customize the format of the tooltip in layers, markers, and bubbles of Maps
- [textStyle](#) - To customize the style of the text in the tooltip for layers, markers, and bubbles of Maps.

INDEX.JS

```
var map = new ej.maps.Maps({
  tooltipRender: function (args) {
    if (!args.options.data) {
      args.cancel = true;
    }
  },
  layers: [{
    shapeData: world_map,
    shapePropertyPath: 'name',
    shapeDataPath: 'name',
    shapeSettings: {
      fill: '#E5E5E5',
      colorMapping: [
        { color: '#b3daff', value: '1' },
        { color: '#80c1ff', value: '2' },
        { color: '#1a90ff', value: '3' },
        { color: '#005cb3', value: '7' }
      ],
      colorValuePath: 'value1'
    },
    dataSource: [
      { "name": "India", "value1": "3", "value2": "2", "country":
"India" },
      { "name": "Dominican Rep.", "value1": "3", "value2": "2",
"country": "West Indies" },
      { "name": "Cuba", "value1": "3", "value2": "2", "country": "West
Indies" },
      { "name": "Jamaica", "value1": "3", "value2": "2", "country":
"West Indies" },
      { "name": "Haiti", "value1": "3", "value2": "2", "country":
"West Indies" },
      { "name": "Guyana", "value1": "3", "value2": "2", "country":
"West Indies" },
      { "name": "Suriname", "value1": "3", "value2": "2", "country":
"West Indies" },
      { "name": "Trinidad and Tobago", "value1": "3", "value2": "2",
"country": "West Indies" },
      { "name": "Sri Lanka", "value1": "3", "value2": "1", "country":
"Sri Lanka" },
      { "name": "United Kingdom", "value1": "3", "value2": "0",
"country": "England" },
      { "name": "Pakistan", "value1": "2", "value2": "1", "country":
"Pakistan" },
      { "name": "New Zealand", "value1": "1", "value2": "0",
"country": "New Zealand" },
      { "name": "Australia", "value1": "7", "value2": "5", "country":
"Australia" }
    ],
    tooltipSettings: {
      visible: true,

```

```

        valuePath: 'name',
        format: '${name}: ${value1}',
        fill: '#D0D0D0',
        textStyle: {
            color: 'green',
            fontFamily: 'Times New Roman',
            fontStyle: 'Sans-serif'
        }
    }
}
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Tooltip template

The HTML element can be rendered in the tooltip of the Maps using the [template](#) property of the [tooltipSettings](#). In the following example, \${value1} and \${value2} are the place holders in the HTML element to display the value1 and value2 values of the corresponding shape.

INDEX.JS

```

var map = new ej.maps.Maps({
    tooltipRender(args) {
        if (!args.options['data']) {
            args.cancel = true;

```



```

    },
    legendSettings: {
        visible: true,
        mode: 'Interactive',
        position: 'Left',
        orientation: 'Vertical',
        height: '70%',
        width: '10'
    },
    layers: [{
        shapeData: world_map,
        shapePropertyPath: 'name',
        shapeDataPath: 'name',
        dataSource: tooltipData,
        tooltipSettings: {
            visible: true,
            valuePath: 'name',
            template: '#template'
        },
        shapeSettings: {
            fill: '#E5E5E5',
            colorMapping: [
                { color: '#b3daff', value: '1' },
                { color: '#80c1ff', value: '2' },
                { color: '#1a90ff', value: '3' },
                { color: '#005cb3', value: '7' }
            ],
            colorValuePath: 'value1'
        }
    ]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="tooltip-datasource.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <div id="template" style="display:none">

```

```

        <div class="toolbar">
            <div class="listing2">
                <center>
                    ${country}
                </center>
            </div>
            <hr style="margin-top: 2px;margin-bottom:5px;border:0.5px solid
#DDDDDD">
            <div>
                <span class="listing1">Finalist : </span><span
class="listing2">${value1}</span>
            </div>
            <div>
                <span class="listing1">Win : </span><span
class="listing2">${value2}</span>
            </div>
        </div>
    </div>
</div>
<style>
    .toolbar {
        border-radius: 4px;
        border: 1px #abb9c6;
        opacity: 90%;
        background: rgba(53, 63, 76, 0.90);
        box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.40);
        padding-bottom: 5px;
        padding-top: 10px;
        padding-left: 10px;
        padding-right: 10px;
        width: 90px;
    }
    .listing1 {
        font-size:13px;
        color:#cccccc
    }
    .listing2 {
        font-size:13px;
        color:#ffffff;
        font-weight: 500;
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Center position zooming](#)
- [Tooltip for marker](#)
- [Navigating to particular country](#)

Print in EJ2 JavaScript Maps control

Print

The rendered Maps can be printed directly from the browser by calling the [print](#) method. To use the print functionality in Maps, set the [allowPrint](#) property to **true**.

INDEX.JS

```
var maps = new ej.maps.Maps({
  allowPrint: true,
  layers: [
    {
      dataLabelSettings: {
        visible: true,
        labelPath: 'name',
        smartLabelMode: 'Trim'
      },
      shapeData: usa_map,
      shapeSettings: {
        autofill: true
      },
      tooltipSettings: {
        visible: true,
        valuePath: 'name'
      },
    },
  ],
});
maps.appendTo('#element');
document.getElementById('print').onclick = () => {
  maps.print();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="usa.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>

    <button id="print" type="button" width="15%" style="float:
right">Print</button>
```

```

</div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export

Image Export

To use the image export functionality, set the [allowImageExport](#) property to **true**. The rendered Maps can be exported as an image using the [export](#) method. The method requires two parameters: image type and file name. The Maps can be exported as an image in the following formats.

- JPEG
- PNG
- SVG

INDEX.JS

```

var maps = new ej.maps.Maps({
    allowImageExport: true,
    layers: [
        {
            dataLabelSettings: {
                visible: true,
                labelPath: 'name',
                smartLabelMode: 'Trim'
            },
            shapeData: usa_map,
            shapeSettings: {
                autofill: true
            },
            tooltipSettings: {
                visible: true,
                valuePath: 'name'
            },
        }
    ]
});
maps.appendTo('#element');
document.getElementById('export').onclick = () => {
    maps.export('PNG', 'Maps');
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Maps</title>
<meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="usa.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 125px">
        <div id="element"></div>
        <button id="export" type="button" width="15%" style="float:
right">Export</button>
        <div id="data"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting Maps as base 64 string of the file

The image can be exported as base64 string for the JPEG and PNG formats. The rendered Maps can be exported to image as a base64 string using the [export](#) method. There are four parameters required: image type, file name, orientation of the exported PDF document which must be set as **null** for image export and finally **allowDownload** which should be set as **false** to return base64 string.

INDEX.JS

```

var maps = new ej.maps.Maps({
    allowImageExport: true,
    layers: [
        {
            dataLabelSettings: {
                visible: true,
                labelPath: 'name',
                smartLabelMode: 'Trim'
            },
            shapeData: usa_map,
            shapeSettings: {
                autofill: true
            },
            tooltipSettings: {
                visible: true,
                valuePath: 'name'
            }
        }
    ]
})

```

```

    });
    maps.appendTo('#element');
    document.getElementById('export').onclick = () => {
        maps.export('JPEG', 'Maps', null, false).then((data) => {
            document.getElementById('data').innerHTML = data;
        });
    };
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="usa.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 125px">
        <div id="element"></div>
        <button id="export" type="button" width="15%" style="float:
right">Export</button>
        <div id="data"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

PDF Export

To use the PDF export functionality, set the [allowPdfExport](#) property to **true**. The rendered Maps can be exported as PDF using the [export](#) method. The [export](#) method requires three parameters: file type, file name and orientation of the PDF document. The orientation setting is optional and **0** indicates portrait and **1** indicates landscape.

INDEX.JS

```

var maps = new ej.maps.Maps({
    allowPdfExport: true,
    layers: [
        {
            dataLabelSettings: {

```

```

        visible: true,
        labelPath: 'name',
        smartLabelMode: 'Trim'
    },
    shapeData: usa_map,
    shapeSettings: {
        autofill: true
    },
    tooltipSettings: {
        visible: true,
        valuePath: 'name'
    }
}
});
maps.appendTo('#element');
document.getElementById('export').onclick = () => {
    maps.export('PDF', 'Maps', 0);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="usa.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
    <button id="export" type="button" width="15%" style="float:
right">Export</button>
    <div id="data"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

The exporting of the Maps as base64 string is not supported for the PDF export.

<!-- markdownlint-disable MD010 -->

Export the tile Maps

The rendered Maps with providers such as OSM, Bing and Google static Maps can be exported using the [export](#) method. It supports the following export formats.

- JPEG
- PNG
- PDF

INDEX.JS

```
var maps = new ej.maps.Maps({
  allowImageExport: true,
  allowPdfExport: true,
  allowPrint: true,
  titleSettings:{
    text:'OSM'
  },
  layers: [
    {
      urlTemplate:"https://tile.openstreetmap.org/level/tileX/tileY.png"
    }
  ]
});
maps.appendTo('#container');
document.getElementById('export').onclick = () => {
  maps.export('PNG', 'Maps');
};
document.getElementById('print').onclick = () => {
  maps.print();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="width:600px;height:500px">
  </div>
  <button id="print" type="button">Print</button><br><br>
```



```

<button id="export" type="button">Export</button>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in EJ2 JavaScript Maps control

Maps has built-in accessibility features like screen reading, keyboard navigation, and WAI-ARIA attributes.

WAI-ARIA attributes

To meet accessibility standards, the Maps control follows to the [WAI-ARIA](#) patterns. In the Maps control, the following ARIA attributes are used:

Attributes	Purpose
---	---
role=region	It specifies the Maps areas that do not support interactive functions like selection and highlight.
role=button	It specifies the Maps areas where interactive functions such as selection and highlight are available.
aria-label	Provides an accessible name for Maps elements such as geometric map shapes, legend title, data labels, and so on. To learn more, see the next topic.

Screen reading in Maps

Accessibility in the Maps control ensures that all users, regardless of ability or disability, can use screen reading. The following Map elements will be read aloud using screen reading software, such as Narrator for Windows.

Elements	Description
---	---
Shapes in the layer	Reads the names of the geographical shapes (such as countries, states, and regions) that appear on the Maps.
Legend title	Reads the contents of the legend's title as specified in Maps.
Legend item label	Reads the label of a legend item in Maps.
Data label	Reads the label specified for the shapes in the Maps layer.
Annotation	Reads the content specified in the annotation.
Marker template	Reads the content provided in the marker template.
Tooltip template	Reads the content provided in the tooltip template.
Data label template	Reads the content provided in the data label template.

Keyboard Navigation

All the Maps actions can be controlled via keyboard keys. The applicable key combinations and their relative functionalities are listed below for the appropriate UI features available in the control.

Interaction Keys | Description

Tab | Moves to the next focusable element on the map, such as the legend or shape.

Shift + Tab | Moves to the previous focusable element on the map, such as the legend or shape.

+ | When zooming is enabled, zoom in operation can be performed.

- | When zooming is enabled, zoom out operation can be performed.

Left arrow | When zoomed in, the map can be scrolled to the left.

Right arrow | When zoomed in, the map can be scrolled to the right.

Up arrow | When zoomed in, the map can be scrolled upward.

Down arrow | When zoomed in, the map can be scrolled downward.

R | When zooming is enabled, reset operation can be performed.

Enter | The page can be navigated to the next and previous states in legend. Similarly, the selection can be made while navigating over the shape.

Ensuring accessibility

The Maps control's accessibility levels are ensured using an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Maps control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Maps control with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript controls](#)

Internationalization in EJ2 JavaScript Maps control

Maps provide support for internationalization for the below elements.

- Data label
- Tooltip

For more information about number and date formatter, refer to the [internationalization](#) section.

<!-- markdownlint-disable MD036 -->

Globalization

Globalization is the process of designing and developing a component that works in different cultures/locales. Internationalization library is used to globalize number, date, time values in Maps component using [format](#) property in the [Maps](#).

Numeric Format

The numeric formats such as currency, percentage and so on can be displayed in the tooltip and data labels of the Maps using the [format](#) property in the [Maps](#). In the below example, the tooltip is globalized to **German** culture. When setting the [useGroupingSeparator](#) property as **true**, the numeric text in the Maps separates with the comma separator.

```
`ts
import { Maps, MapsTooltip } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
import { setCulture } from '@syncfusion/ej2-base';
setCulture('de');
Maps.Inject(MapsTooltip);
let maps: Maps = new Maps({
  format: 'c',
  useGroupingSeparator: true,
  layers: [
    {
      shapeData: world_map,
      dataSource: [
        { "Country": "China", "Membership": "Permanent", population: '38332521' },
        { "Country": "France", "Membership": "Permanent", population: '19651127' },
        { "Country": "Russia", "Membership": "Permanent", population: '3090416' },
        { "Country": "Kazakhstan", "Membership": "Non-Permanent", population: '1232521' },
        { "Country": "Poland", "Membership": "Non-Permanent", population: '90332521' },
        { "Country": "Sweden", "Membership": "Non-Permanent", population: '383521' }
      ],
      shapePropertyPath: 'name',
      shapeDataPath: 'Country',
      shapeSettings: {
        colorValuePath: 'Membership',
        colorMapping: [
          {
            value: 'Permanent', color: '#D84444'
          },
          {
```

```

value: 'Non-Permanent', color: '#316DB5'
}}
},
tooltipSettings: {
visible: true,
valuePath: 'population'
}
}
]
});
maps.appendTo('#element');
`

```

Localization in EJ2 JavaScript Maps control

The localization library allows localizing the default text content of the Maps control. The Maps control has the static text of some features such as tooltip of zoom toolbar, and that can be changed to any other culture(Arabic, Deutsch, French, etc) by defining the locale value and translation object.

<!-- markdownlint-disable MD033 -->

Locale key words	Text to display
Zoom	Zoom
ZoomIn	Zoom In
ZoomOut	Zoom Out
Reset	Reset
Pan	Pan

To load translation object in the application, use `load` function of **L10n** class. For more information about localization, refer [here](#).

INDEX.TS

```

import { Maps, Zoom } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
import { L10n } from '@syncfusion/ej2-base';
Maps.Inject(Zoom);
L10n.load({
  'ar-AR': {
    'maps': {
      ZoomIn: 'تكبير',
      ZoomOut: 'تصغير',
      Zoom: 'زوم',
      Pan: 'مقللة',

```

```

        Reset: 'إعادة تعيين'
      },
    },
  });
let map: Maps = new Maps({
  zoomSettings: {
    enable: true,
    toolbars: ['ZoomIn', 'ZoomOut', 'Zoom', 'Pan', 'Reset' ]
  },
  locale: 'ar-AR',
  layers: [
    {
      shapeData: world_map
    }
  ]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Methods in EJ2 JavaScript Maps control

Methods

This section explains the methods used in the Maps control.

getMinMaxLatitudeLongitude

The `getMinMaxLatitudeLongitude` method returns the minimum and maximum latitude and longitude values of the Maps visible area. This method returns a [IMinMaxLatitudeLongitude](#) object that contains the Maps minimum and maximum latitude and longitude coordinates.

INDEX.JS

```
var maps = new ej.maps.Maps({
  zoomSettings: {
    enable: true,
    zoomFactor: 7
  },
  centerPosition: {
    latitude: 21.815447,
    longitude: 80.1932
  },
  layers: [
    {
      shapeData: world_map,
      markerSettings: [
        {
          visible: true,
          height: 25,
          width: 25,
          shape: 'Circle',
          animationDuration: 1500,
          dataSource: [
            {
              latitude: 22.572646,
              longitude: 88.363895
            },
            {
              latitude: 25.0700428,
              longitude: 67.2847875
            }
          ]
        }
      ]
    }
  ]
});
maps.appendTo('#element');
function formatKey(key) {
  if (key === 'minLatitude') {
    return 'Minimum Latitude';
  } else if (key === 'maxLatitude') {
    return 'Maximum Latitude';
  } else if (key === 'minLongitude') {
    return 'Minimum Longitude';
  } else if (key === 'maxLongitude') {
    return 'Maximum Longitude';
  }
}
document.getElementById('button').onclick = () => {
  var mapBoundCoordinates;
  mapBoundCoordinates = maps.getMinMaxLatitudeLongitude();
}
```

```

const displayDiv = document.getElementById('coordinatesDisplay');
displayDiv.innerHTML = '';
if (mapBoundCoordinates) {
    for (const key in mapBoundCoordinates) {
        if (Object.hasOwnProperty.call(mapBoundCoordinates, key)) {
            const p = document.createElement('p');
            const formattedKey = formatKey(key);
            p.textContent = `${formattedKey}:`;
            displayDiv.appendChild(p);
        }
    }
} else {
    displayDiv.textContent = 'No coordinates available';
}
};

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="margin-top: 125px">
        <button id="button" width="15%">GetMinMaxLatitudeLongitude</button>
        <p id="coordinatesDisplay"></p>
        <div id="element"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Ej1 api migration in EJ2 JavaScript Maps control

This article describes the API migration process of Maps component from Essential JS 1 to Essential JS 2.

Size Customization

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Height | Not Applicable | **Property:** *height*
var maps = new ej.maps.Maps({
height : '300px' }); maps.appendTo('#container');

| Width | Not Applicable | **Property:** *width*
var maps = new ej.maps.Maps({
width : '600px' }); maps.appendTo('#container');

Title and Subtitle Customization

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Title Text | Not Applicable | **Property:** *title.text*
var maps = new ej.maps.Maps({
titleSettings:{ text : 'Members of the UN Security Council' } }); maps.appendTo('#container');

| Subtitle Text | Not Applicable | **Property:** *title.subtitle.text*
var maps = new ej.maps.Maps({
titleSettings:{ subtitleSettings:{ text : 'In 2017' } } }); maps.appendTo('#container');

| Title Alignment | Not Applicable | **Property:** *title.alignment*
var maps = new ej.maps.Maps({
titleSettings:{ alignment : 'Center' } }); maps.appendTo('#container');

| Subtitle Alignment | Not Applicable | **Property:** *title.subtitle.alignment*
var maps = new ej.maps.Maps({
titleSettings:{ subtitleSettings:{ alignment : 'Center' } } }); maps.appendTo('#container');

<!-- markdownlint-disable MD034 -->

Layer Customization

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Type | Not Applicable | **Property:** *layers.type*
var maps = new ej.maps.Maps({
layers:[{ type:'Layer' }] }); maps.appendTo('#container');

| Layer Type | **Property:** *layers.layerType*
\$("#container").ejMap({
layers:[{ layerType:'geometry' }] }); | **Property:** *layers.layerType*
var maps = new ej.maps.Maps({
layers:[{ layerType:'Geometry' }] }); maps.appendTo('#container');

| Visible | Not Applicable | **Property:** *layers.visible*
var maps = new ej.maps.Maps({
layers:[{ visible:true }] }); maps.appendTo('#container');

| Bing Map Type | **Property:** *layers.bingMapType*
\$("#container").ejMap({
layers:[{ bingMapType:'aerial' }] }); | **Property:** *layers.bingMapType*
var maps = new ej.maps.Maps({
layers:[{ bingMapType:'Aerial' }] }); maps.appendTo('#container');

| Bing Map Key | **Property:** *layers.key*
\$("#container").ejMap({
layers:[{ key:'' }] }); | **Property:** *layers.key*
var maps = new ej.maps.Maps({
layers:[{ key:'' }] }); maps.appendTo('#container');

| URL Template | **Property:** *layers.urlTemplate*
\$("#container").ejMap({
layers:[{ urlTemplate:'http://a.tile.openstreetmap.org/level/tileX/tileY.png' }] }); | **Property:** *layers.urlTemplate*
var maps = new ej.maps.Maps({
layers:[{


```
urlTemplate:'http://a.tile.openstreetmap.org/level/tileX/tileY.png' }} <br/> }}; <br/>
maps.appendTo('#container')|
```

```
| Shape Data | Property: layers.shapeData<br/><br/> $("#container").ejMap({ <br/> &#160; layers:[{
shapeData:'WorldMap' }} <br/> }}; | Property: layers.shapeData<br/><br/> var maps = new
ej.maps.Maps({ <br/> &#160; layers:[{ shapeData:'WorldMap' }} <br/> }}; <br/>
maps.appendTo('#container')|
```

```
| Data Source | Property: layers.dataSource<br/><br/> $("#container").ejMap({ <br/> &#160; layers:[{
dataSource:'PopulationData' }} <br/> }}; | Property: layers.dataSource<br/><br/> var maps = new
ej.maps.Maps({ <br/> &#160; layers:[{ dataSource:'PopulationData' }} <br/> }}; <br/>
maps.appendTo('#container')|
```

```
| Query | Not Applicable | Property: layers.query<br/><br/> var maps = new ej.maps.Maps({ <br/>
&#160; layers:[{ query:'' }} <br/> }}; <br/> maps.appendTo('#container')|
```

```
| Shape Data Path | Property: layers.shapeDataPath<br/><br/> $("#container").ejMap({ <br/> &#160;
layers:[{ shapeDataPath:'Continent' }} <br/> }}; | Property: layers.shapeDataPath<br/><br/> var maps =
new ej.maps.Maps({ <br/> &#160; layers:[{ shapeDataPath:'Continent' }} <br/> }}; <br/>
maps.appendTo('#container')|
```

```
| Shape Property Path | Property: layers.shapePropertyPath<br/><br/> $("#container").ejMap({ <br/>
&#160; layers:[{ shapePropertyPath:'Continent' }} <br/> }}; | Property:
layers.shapePropertyPath<br/><br/> var maps = new ej.maps.Maps({ <br/> &#160; layers:[{
shapePropertyPath:'Continent' }} <br/> }}; <br/> maps.appendTo('#container')|
```

```
| Layer Animation | Not Applicable | Property: layers.animationDuration<br/><br/> var maps = new
ej.maps.Maps({ <br/> &#160; layers:[{ animationDuration: 100 }} <br/> }}; <br/>
maps.appendTo('#container')|
```

Shape Customization

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

```
| Shape Fill | Property: layers.shapeSettings.fill<br/><br/> $("#container").ejMap({ <br/> &#160;
layers:[{ shapeSettings:{ fill:'#626171' }} <br/> }}; | Property: layers.shapeSettings.fill<br/><br/> var
maps = new ej.maps.Maps({ <br/> &#160; layers:[{ shapeSettings:{ fill: '#626171' }} <br/> }}; <br/>
maps.appendTo('#container')|
```

```
| Shape Palette | Property: layers.shapeSettings.colorPalette<br/><br/> $("#container").ejMap({ <br/>
&#160; layers:[{ shapeSettings:{ colorPalette:'customPalette' }} <br/> }}; | Property:
layers.shapeSettings.palette<br/><br/> var maps = new ej.maps.Maps({ <br/> &#160; layers:[{
shapeSettings:{ palette: ['red','green'] }} <br/> }}; <br/> maps.appendTo('#container')|
```

```
| Shape Point Radius | Not Applicable | Property: layers.shapeSettings.circleRadius<br/><br/> var maps
= new ej.maps.Maps({ <br/> &#160; layers:[{ shapeSettings:{ circleRadius: 10 }} <br/> }}; <br/>
maps.appendTo('#container')|
```

```
| Shape Color Value Path | Property: layers.shapeSettings.colorValuePath<br/><br/>
$("#container").ejMap({ <br/> &#160; layers:[{ shapeSettings:{ colorValuePath:'Country' }} <br/> }}; |
Property: layers.shapeSettings.colorValuePath<br/><br/> var maps = new ej.maps.Maps({ <br/> &#160;
layers:[{ shapeSettings:{ colorValuePath:'Country' }} <br/> }}; <br/> maps.appendTo('#container')|
```

| Shape Value Path | **Property:** *layers.shapeSettings.valuePath*
 \$("#container").ejMap({
 layers:[{ shapeSettings:{ valuePath:'population' } }]
 layers.shapeSettings.valuePath
 var maps = new ej.maps.Maps({
 shapeSettings:{ valuePath:'population' } }]
 maps.appendTo('#container')|

| Shape DashArray | Not Applicable | **Property:** *layers.shapeSettings.dashArray*
 var maps = new ej.maps.Maps({
 layers:[{ shapeSettings:{ dashArray:'1,2' } }]
 maps.appendTo('#container')|

| Shape Opacity | Not Applicable | **Property:** *layers.shapeSettings.opacity*
 var maps = new ej.maps.Maps({
 layers:[{ shapeSettings:{ opacity:'0.5' } }]
 maps.appendTo('#container')|

| Range Color Mapping | **Property:** *layers.shapeSettings.colorMappings.rangeColorMapping*
 \$("#container").ejMap({
 layers:[{ shapeSettings:{ colorMappings:{ rangeColorMapping:[{
 from:'10', to:'40', color: "#D84444" } }] } }]
 layers.shapeSettings.colorMapping
 var maps = new ej.maps.Maps({
 shapeSettings:{ colorMapping:[{ from:10, to: 30, color: "#D84444" } }] }]
 maps.appendTo('#container')|

| Equal Color Mapping | **Property:** *layers.shapeSettings.colorMappings.equalColorMapping*
 \$("#container").ejMap({
 layers:[{ shapeSettings:{ colorMappings:{ equalColorMapping:[{
 value: "Romney", color: "#D84444" } }] } }]
 layers.shapeSettings.colorMapping
 var maps = new ej.maps.Maps({
 shapeSettings:{ colorMapping:[{ value: 'Romney', color: '#D84444' } }] }]
 maps.appendTo('#container')|

Marker Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Marker Data Source | **Property:** *layers.markers*
 \$("#container").ejMap({
 layers:[{ markers:[{ "Name": "USA", "latitude": 38.8833, "longitude": -77.0167 } }] }]
 layers.markerSettings.dataSource
 var maps = new ej.maps.Maps({
 markerSettings:{ dataSource: [{ "Name": "USA", "latitude": 38.8833, "longitude": -77.0167 } }] }]
 maps.appendTo('#container')|

| Marker Template | **Property:** *layers.markerTemplate*
 \$("#container").ejMap({
 layers:[{ markerTemplate:"Template" }] }]
 layers.markerSettings.template
 var maps = new ej.maps.Maps({
 layers:[{ markerSettings:{ template:'Template' } }] }]
 maps.appendTo('#container')|

| Marker Visible | Not Applicable | **Property:** *layers.markerSettings.visible*
 var maps = new ej.maps.Maps({
 layers:[{ markerSettings:{ visible:true } }] }]
 maps.appendTo('#container')|

| Marker Fill | Not Applicable | **Property:** *layers.markerSettings.fill*
 var maps = new ej.maps.Maps({
 layers:[{ markerSettings:{ fill:'#FF471A' } }] }]
 maps.appendTo('#container')|

| Marker Height | Not Applicable | **Property:** *layers.markerSettings.height*
 var maps = new ej.maps.Maps({
 layers:[{ markerSettings:{ height:20 } }] }]
 maps.appendTo('#container')|

| Marker Width | Not Applicable | **Property:** *layers.markerSettings.width*
 ej.maps.Maps({
 layers:[{ markerSettings:{ width:20 } }]
 });

 maps.appendTo('#container')|

| Marker Shape | Not Applicable | **Property:** *layers.markerSettings.shape*
 ej.maps.Maps({
 layers:[{ markerSettings:{ shape:'Balloon' } }]
 });

 maps.appendTo('#container')|

| Marker ImageURL | Not Applicable | **Property:** *layers.markerSettings.imageUrl*
 new ej.maps.Maps({
 layers:[{ markerSettings:{
 imageUrl:'http://js.syncfusion.com/demos/web/Images/map/pin.png' } }]
 });

 maps.appendTo('#container')|

| Marker Opacity | Not Applicable | **Property:** *layers.markerSettings.opacity*
 ej.maps.Maps({
 layers:[{ markerSettings:{ opacity:0.5 } }]
 });

 maps.appendTo('#container')|

| Marker Legend Text | Not Applicable | **Property:** *layers.markerSettings.legendText*
 maps = new ej.maps.Maps({
 layers:[{ markerSettings:{ legendText:'China' } }]
 });

 maps.appendTo('#container')|

| Marker Offset | Not Applicable | **Property:** *layers.markerSettings.offset*
 ej.maps.Maps({
 layers:[{ markerSettings:{ offset:new Point(20, 20) } }]
 });

 maps.appendTo('#container')|

| Marker Animation Duration | Not Applicable | **Property:**
layers.markerSettings.animationDuration
 ej.maps.Maps({
 layers:[{ markerSettings:{ animationDuration:2000 } }]
 });

 maps.appendTo('#container')|

| Marker Animation Delay | Not Applicable | **Property:**
layers.markerSettings.animationDelay
 ej.maps.Maps({
 layers:[{ markerSettings:{ animationDelay:100 } }]
 });

 maps.appendTo('#container')|

| Marker DashArray | Not Applicable | **Property:** *layers.markerSettings.dashArray*
 new ej.maps.Maps({
 layers:[{ markerSettings:{ dashArray:'2,3' } }]
 });

 maps.appendTo('#container')|

| Marker Selection | Not Applicable | **Property:** *layers.markerSettings.selectionSettings*
 maps = new ej.maps.Maps({
 layers:[{ markerSettings:{

 selectionSettings : { enable:true,fill:'#D2691E',opacity:1,enableMultiSelect:false }
 } }]

 });
 maps.appendTo('#container')|

| Marker Highlight | Not Applicable | **Property:** *layers.markerSettings.highlightSettings*
 maps = new ej.maps.Maps({
 layers:[{ markerSettings:{

 highlightSettings : { enable:true,fill:'#D2691E',opacity:1 }
 } }]
 });

 maps.appendTo('#container')|

| Marker Tooltip | Not Applicable | **Property:** *layers.markerSettings.tooltipSettings*
 new ej.maps.Maps({
 layers:[{ markerSettings:{

 tooltipSettings : { visible:true,fill:'#363F4C',template:'TooltipTemplate', valuePath:'State',
 format:'\${State}
\${District}' }
 } }]
 });
 maps.appendTo('#container')|

Bubble Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Visible | **Property:** *layers.bubbleSettings.visible*
 layers:{{ bubbleSettings: { showBubble:true } }} | **Property:**
layers.bubbleSettings.visible var maps = new ej.maps.Maps({ layers:{{
 bubbleSettings:{ visible:true } }}; maps.appendTo('#container')

| ValuePath | **Property:** *layers.bubbleSettings.valuePath*
 layers:{{ bubbleSettings: { valuePath:'Population' } }} | **Property:**
layers.bubbleSettings.valuePath var maps = new ej.maps.Maps({ layers:{{
 bubbleSettings:{ valuePath:'Population' } }}; maps.appendTo('#container')

| MinValue | **Property:** *layers.bubbleSettings.minValue*
 layers:{{ bubbleSettings: { minValue:20 } }} | **Property:**
layers.bubbleSettings.minRadius var maps = new ej.maps.Maps({ layers:{{
 bubbleSettings:{ minRadius:10 } }}; maps.appendTo('#container')

| MaxValue | **Property:** *layers.bubbleSettings.maxValue*
 layers:{{ bubbleSettings: { maxValue:30 } }} | **Property:**
layers.bubbleSettings.maxRadius var maps = new ej.maps.Maps({ layers:{{
 bubbleSettings:{ maxRadius:20 } }}; maps.appendTo('#container')

| Bubble Type | Not Applicable | **Property:** *layers.bubbleSettings.bubbleType*
 var maps = new ej.maps.Maps({ layers:{{ bubbleSettings:{ bubbleType:'Circle' } }}; maps.appendTo('#container')

| Color | **Property:** *layers.bubbleSettings.color*
 layers:{{ bubbleSettings: { color:'green' } }} | **Property:** *layers.bubbleSettings.fill*
 var maps = new ej.maps.Maps({ layers:{{ bubbleSettings:{ fill:'red' } }}; maps.appendTo('#container')

| Opacity | **Property:** *layers.bubbleSettings.bubbleOpacity*
 layers:{{ bubbleSettings: { bubbleOpacity:0.5 } }} | **Property:**
layers.bubbleSettings.opacity var maps = new ej.maps.Maps({ layers:{{
 bubbleSettings:{ opacity:0.5 } }}; maps.appendTo('#container')

| Color Value Path | **Property:** *layers.bubbleSettings.colorValuePath*
 layers:{{ bubbleSettings: { colorValuePath:'Population' } }} | **Property:**
layers.bubbleSettings.colorValuePath var maps = new ej.maps.Maps({ layers:{{
 bubbleSettings:{ colorValuePath:'Population' } }}; maps.appendTo('#container')

| Enable Tooltip | **Property:** *layers.bubbleSettings.showTooltip*
 layers:{{ bubbleSettings: { showTooltip:true } }} | **Property:**
layers.bubbleSettings.tooltipSettings.visible var maps = new ej.maps.Maps({ layers:{{
 bubbleSettings:{{ tooltipSettings: { visible:true } }}; maps.appendTo('#container')

| Tooltip Template | **Property:** *layers.bubbleSettings.tooltipTemplate*
 layers:{{ bubbleSettings: { tooltipTemplate:'Template' } }} | **Property:** *layers.bubbleSettings.tooltipSettings.template*
 var maps = new ej.maps.Maps({ layers:{{ bubbleSettings:{{ tooltipSettings: {
 template:'TooltipTemplate' } }}; maps.appendTo('#container')

| Bubble Selection | Not Applicable | **Property:** *layers.bubbleSettings.selectionSettings*
 maps = new ej.maps.Maps({ layers:[{ bubbleSettings:{
 selectionSettings : { enable:true,fill:'#D2691E',opacity:1,enableMultiSelect:false }
 }
 }
 }); maps.appendTo('#container')

| Bubble Highlight | Not Applicable | **Property:** *layers.bubbleSettings.highlightSettings*
 maps = new ej.maps.Maps({ layers:[{ bubbleSettings:{
 highlightSettings : { enable:true,fill:'#D2691E',opacity:1 }
 }
 }
 }); maps.appendTo('#container')

| Range Color Mapping | **Property:** *layers.bubbleSettings.colorMappings.rangeColorMapping*
 \$("#container").ejMap({ layers:[{ bubbleSettings:{ colorMappings:{ rangeColorMapping:{
 from:'10', to:'40', color: "#D84444" }
 }
 }
 }
 }
 }); | **Property:**
layers.bubbleSettings.colorMapping
 maps = new ej.maps.Maps({ layers:[{
 bubbleSettings:{ colorMapping:[{ from:10, to: 30, color: "#D84444" }
]
 }
 }
 }); maps.appendTo('#container')

| Equal Color Mapping | **Property:** *layers.bubbleSettings.colorMappings.equalColorMapping*
 \$("#container").ejMap({ layers:[{ bubbleSettings:{ colorMappings:{ equalColorMapping:[{
 value: "Romney", color: "#D84444" }
]
 }
 }
 }
 }); | **Property:**
layers.bubbleSettings.colorMapping
 maps = new ej.maps.Maps({ layers:[{
 bubbleSettings:{ colorMapping:[{ value: 'Romney', color: '#D84444' }
]
 }
 }
 }); maps.appendTo('#container')

DataLabel Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Visible | **Property:** *layers.labelSettings.showLabels*
 \$("#container").ejMap({ layers:[{ labelSettings: { showLabels:true }
 }
 }
 }); | **Property:**
layers.dataLabelSettings.visible
 maps = new ej.maps.Maps({ layers:[{
 dataLabelSettings:{ visible:true }
 }
 }
 }); maps.appendTo('#container')

| Label Path | **Property:** *layers.labelSettings.labelPath*
 \$("#container").ejMap({ layers:[{ labelSettings: { labelPath:'Continent' }
 }
 }
 }); | **Property:**
layers.dataLabelSettings.labelPath
 maps = new ej.maps.Maps({ layers:[{
 dataLabelSettings:{ labelPath:'Continent' }
 }
 }
 }); maps.appendTo('#container')

| Enable Smart Label | **Property:** *layers.labelSettings.enableSmartLabel*
 \$("#container").ejMap({ layers:[{ labelSettings: { enableSmartLabel:true }
 }
 }
 }); | Not Applicable |

| Smart Label Size | **Property:** *layers.labelSettings.smartLabelSize*
 \$("#container").ejMap({ layers:[{ labelSettings: { smartLabelSize:20 }
 }
 }
 }); | Not Applicable |

| Label Length | **Property:** *layers.labelSettings.labelLength*
 \$("#container").ejMap({ layers:[{ labelSettings: { labelLength:20 }
 }
 }
 }); | Not Applicable |

| Opacity | Not Applicable | **Property:** *layers.dataLabelSettings.opacity*
 maps = new ej.maps.Maps({ layers:[{ dataLabelSettings:{ opacity:0.5 }
 }
 }
 }); maps.appendTo('#container')

| Smart Label Mode | Not Applicable | **Property:** *layers.dataLabelSettings.smartLabelMode*

var maps = new ej.maps.Maps({
 layers:[{ dataLabelSettings:{ smartLabelMode:'Trim' } }]

 });
 maps.appendTo('#container')|

| InterSectAction | Not Applicable | **Property:** *layers.dataLabelSettings.intersectionAction*

 var
maps = new ej.maps.Maps({
 layers:[{ dataLabelSettings:{ intersectionAction:'Trim' } }]
});
 maps.appendTo('#container')|

| Template | Not Applicable | **Property:** *layers.dataLabelSettings.template*

 var maps = new
ej.maps.Maps({
 layers:[{ dataLabelSettings:{ template:'LabelTemplate' } }]
});
 maps.appendTo('#container')|

Legend Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Visible | **Property:** *layers.legendSettings.showLegend*

 \$(" #container").ejMap({

layers:[{ legendSettings: { showLegend:true } }]
});
 | **Property:** *legendSettings.visible*

 var
maps = new ej.maps.Maps({
 legendSettings: { visible:true } }]
});
 maps.appendTo('#container')|

| Toggle Visibility | **Property:** *layers.legendSettings.toggleVisibility*

 \$(" #container").ejMap({

 layers:[{ legendSettings: { toggleVisibility:true } }]
});
 | **Property:**
legendSettings.toggleVisibility

 var maps = new ej.maps.Maps({
 legendSettings:
{ toggleVisibility:true } }]
});
 maps.appendTo('#container')|

| Legend Location X | **Property:** *layers.legendSettings.positionX*

 \$(" #container").ejMap({

 layers:[{ legendSettings: { positionX:250 } }]
});
 | **Property:**
legendSettings.location

 var maps = new ej.maps.Maps({
 legendSettings: {
location:new Point(250,0) } }]
});
 maps.appendTo('#container')|

| Legend Location Y | **Property:** *layers.legendSettings.positionY*

 \$(" #container").ejMap({

 layers:[{ legendSettings: { positionY:350 } }]
});
 | **Property:**
legendSettings.location

 var maps = new ej.maps.Maps({
 legendSettings: {
location:new Point(0,350)} }]
});
 maps.appendTo('#container')|

| Legend Type | **Property:** *layers.legendSettings.type*

 \$(" #container").ejMap({

layers:[{ legendSettings: { type:'Layers' } }]
});
 | **Property:** *legendSettings.type*

 var maps =
new ej.maps.Maps({
 legendSettings: { type:'Layers' } }]
});
 maps.appendTo('#container') |

| Label Orientation | **Property:** *layers.legendSettings.labelOrientation*

\$(" #container").ejMap({
 layers:[{ legendSettings: { labelOrientation:'Vertical' } }]
});
 |
Not Applicable |

| Legend Title | **Property:** *layers.legendSettings.title*

 \$(" #container").ejMap({

layers:[{ legendSettings: { title:'Union territories of India' } }]
});
 | **Property:**
legendSettings.title

 var maps = new ej.maps.Maps({
 legendSettings: {
title:'Union territories of India' } }]
});
 maps.appendTo('#container') |

| Legend Mode | **Property:** *layers.legendSettings.mode*

 \$(" #container").ejMap({

 layers:[{ legendSettings: { mode:'Default' } }]
});
 | **Property:**

legendSettings.mode
var maps = new ej.maps.Maps({
mode:'Default'}); maps.appendTo('#container') |

| Legend Position | **Property:** *layers.legendSettings.position*
\$("#container").ejMap({
layers:[{ legendSettings: { position:'TopLeft' } }]} | **Property:**

legendSettings.position
var maps = new ej.maps.Maps({
position:'Top'}); maps.appendTo('#container') |

| Legend DockOnMap | **Property:** *layers.legendSettings.dockOnMap*
\$("#container").ejMap({
layers:[{ legendSettings: { dockOnMap:true } }]} | Not Applicable |

| Legend Alignment | **Property:** *layers.legendSettings.dockPosition*
\$("#container").ejMap({
layers:[{ legendSettings: { dockPosition:'Right' } }]} | **Property:**

legendSettings.alignment
var maps = new ej.maps.Maps({
alignment:'Center'}); maps.appendTo('#container') |

| Legend Left Label | **Property:** *layers.legendSettings.leftLabel*
\$("#container").ejMap({
layers:[{ legendSettings: { leftLabel:'1000M' } }]} | Not Applicable |

| Legend Right Label | **Property:** *layers.legendSettings.rightLabel*
\$("#container").ejMap({
layers:[{ legendSettings: { rightLabel:'3000M' } }]} | Not Applicable |

| Legend Shape | **Property:** *layers.legendSettings.icon*
\$("#container").ejMap({
layers:[{ legendSettings: { icon:'Circle' } }]} | **Property:** *legendSettings.shape*
var maps = new ej.maps.Maps({
legendSettings: { shape:'Circle'}}); maps.appendTo('#container') |

| Legend Shape Height | **Property:** *layers.legendSettings.iconHeight*
\$("#container").ejMap({
layers:[{ legendSettings: { iconHeight: 20 } }]} | **Property:** *legendSettings.shapeHeight*
var maps = new ej.maps.Maps({
shapeHeight:20 }); maps.appendTo('#container') |

| Legend Shape Width | **Property:** *layers.legendSettings.iconWidth*
\$("#container").ejMap({
layers:[{ legendSettings: { iconWidth: 20 } }]} | **Property:** *legendSettings.shapeWidth*
var maps = new ej.maps.Maps({
shapeWidth:20 }); maps.appendTo('#container') |

| Height | **Property:** *layers.legendSettings.height*
\$("#container").ejMap({
layers:[{ legendSettings: { height: 50 } }]} | **Property:** *legendSettings.width*
var maps = new ej.maps.Maps({
height:'50' }); maps.appendTo('#container') |

| Width | **Property:** *layers.legendSettings.width*
\$("#container").ejMap({
layers:[{ legendSettings: { width: 150 } }]} | **Property:** *legendSettings.width*
var maps = new ej.maps.Maps({
width:'150' }); maps.appendTo('#container') |

| Show Labels | **Property:** *layers.legendSettings.showLabels*
\$("#container").ejMap({
layers:[{ legendSettings: { showLabels: true } }]} | Not Applicable |

| Background | Not Applicable | **Property:** *legendSettings.background*
var maps = new ej.maps.Maps({
legendSettings: { background:'transparent' } }); maps.appendTo('#container') |

| Label Position | Not Applicable | **Property:** *legendSettings.labelPosition*
 var maps = new ej.maps.Maps({
 legendSettings: { labelPosition:'After' } }

 maps.appendTo('#container') |

| Label Display Mode | Not Applicable | **Property:** *legendSettings.labelDisplayMode*
 var maps = new ej.maps.Maps({
 legendSettings: { labelDisplayMode:'Trim' } }

 maps.appendTo('#container') |

| Label Display Mode | Not Applicable | **Property:** *legendSettings.labelDisplayMode*
 var maps = new ej.maps.Maps({
 legendSettings: { labelDisplayMode:'Trim' } }

 maps.appendTo('#container') |

| Legend Orientation | Not Applicable | **Property:** *legendSettings.orientation*
 var maps = new ej.maps.Maps({
 legendSettings: { orientation:'Horizontal' } }

 maps.appendTo('#container') |

| Legend Item Fill | Not Applicable | **Property:** *legendSettings.fill*
 var maps = new ej.maps.Maps({
 legendSettings: { fill:'red' } }

 maps.appendTo('#container') |

| Legend Shape Padding | Not Applicable | **Property:** *legendSettings.shapePadding*
 var maps = new ej.maps.Maps({
 legendSettings: { shapePadding:20 } }

 maps.appendTo('#container') |

| Legend Shape Border Color | Not Applicable | **Property:** *legendSettings.shapeBorder.color*
 var maps = new ej.maps.Maps({
 legendSettings: { shapeBorder:{ color:'green' } } }

 maps.appendTo('#container') |

| Legend Shape Border Width | Not Applicable | **Property:** *legendSettings.shapeBorder.width*
 var maps = new ej.maps.Maps({
 legendSettings: { shapeBorder:{ width:2 } } }

 maps.appendTo('#container') |

| Inverter Pointer | Not Applicable | **Property:** *legendSettings.invertedPointer*
 var maps = new ej.maps.Maps({
 legendSettings: { invertedPointer: true } }

 maps.appendTo('#container') |

| Item Text Style | Not Applicable | **Property:** *legendSettings.textStyle*
 var maps = new ej.maps.Maps({
 legendSettings: { textStyle: { size:'12px' } } }

 maps.appendTo('#container') |

| Title Style | Not Applicable | **Property:** *legendSettings.textStyle*
 var maps = new ej.maps.Maps({
 legendSettings: { textStyle: { size:'12px' } } }

 maps.appendTo('#container') |

Zooming Customization

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Enable | Not Applicable | **Property:** *zoomSettings.enableZoom*
 var maps = new ej.maps.Maps({
 zoomSettings:{ enable:true } }

 maps.appendTo('#container') |

| Minimum Zoom | **Property:** *zoomSettings.minValue*
 \$("container").ejMap({
 zoomSettings:{ minValue:2 } }

 | **Property:** *zoomSettings.minZoom*
 var maps = new ej.maps.Maps({
 zoomSettings:{ minZoom:2 } }

 maps.appendTo('#container') |

| Maximum Zoom | **Property:** *zoomSettings.maxValue*

 \$("#container").ejMap({

zoomSettings:{ maxValue:50 }
 }); | **Property:** *zoomSettings.maxZoom*

 var maps = new
ej.maps.Maps({
 zoomSettings:{ maxZoom:5 }
 });

maps.appendTo('#container')|

| Mouse Wheel Zoom | **Property:** *zoomSettings.enableMouseWheelZoom*

\$("#container").ejMap({
 zoomSettings:{ enableMouseWheelZoom:true }
 }); |
Property: *zoomSettings.maxZoom*

 var maps = new ej.maps.Maps({

zoomSettings:{ mouseWheelZoom:true }
 });
 maps.appendTo('#container')|

| Double Click Zoom | Not Applicable | **Property:** *zoomSettings.doubleClickZoom*

 var maps =
new ej.maps.Maps({
 zoomSettings:{ doubleClickZoom:true }
 });

maps.appendTo('#container')|

| Pinch Zoom | Not Applicable | **Property:** *zoomSettings.pinchZooming*

 var maps = new
ej.maps.Maps({
 zoomSettings:{ pinchZooming:true }
 });

maps.appendTo('#container')|

| Single Click Zoom | **Property:** *zoomSettings.enableZoomOnSelection*

\$("#container").ejMap({
 zoomSettings:{ enableZoomOnSelection:true }
 }); |
Property: *zoomSettings.zoomOnClick*

 var maps = new ej.maps.Maps({

zoomSettings:{ zoomOnClick:true }
 });
 maps.appendTo('#container')|

| Zoom Factor | **Property:** *zoomSettings.factor*

 \$("#container").ejMap({

zoomSettings:{ factor:2 }
 }); | **Property:** *zoomSettings.zoomFactor*

 var maps = new
ej.maps.Maps({
 zoomSettings:{ zoomFactor:2 }
 });

maps.appendTo('#container')|

| Toolbars | Not Applicable | **Property:** *zoomSettings.toolbars*

 var maps = new
ej.maps.Maps({
 zoomSettings:{ toolbars:['ZoomIn'] }
 });

maps.appendTo('#container')|

| Toolbar Orientation | Not Applicable | **Property:** *zoomSettings.toolBarOrientation*

 var maps
= new ej.maps.Maps({
 zoomSettings:{ toolBarOrientation:'Horizontal' }
 });

maps.appendTo('#container')|

| Toolbar Vertical Alignment | Not Applicable | **Property:** *zoomSettings.verticalAlignment*

 var
maps = new ej.maps.Maps({
 zoomSettings:{ verticalAlignment:'Center' }
 });

maps.appendTo('#container')|

| Toolbar Horizontal Alignment | Not Applicable | **Property:**
zoomSettings.horizontalAlignment

 var maps = new ej.maps.Maps({

zoomSettings:{ horizontalAlignment:'Center' }
 });
 maps.appendTo('#container')|

| Toolbar Highlight Color | Not Applicable | **Property:** *zoomSettings.highlightColor*

 var maps
= new ej.maps.Maps({
 zoomSettings:{ highlightColor:'#e61576' }
 });

maps.appendTo('#container')|

| Toolbar Selection Color | Not Applicable | **Property:** *zoomSettings.selectionColor*

 var maps
= new ej.maps.Maps({
 zoomSettings:{ selectionColor:'#e61576' }
 });

maps.appendTo('#container')|

| Toolbar Fill Color | Not Applicable | **Property:** *zoomSettings.color*
 ej.maps.Maps({
 zoomSettings:{ color:'#e61576' }
 });

 maps.appendTo('#container')|

Highlight And Selection Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Highlight Fill | **Property:** *layers.shapeSettings.highlightColor*
 \$("#container").ejMap({
 layers:{ shapeSettings:{ highlightColor:'green' } }
 }); | **Property:** *fill*
 var maps = new ej.maps.Maps({
 layers:[{ highlightSettings:{ fill:'green' }]
 });

 maps.appendTo('#container')|

| Enable Highlight | **Property:** *layers.enableMouseHover*
 \$("#container").ejMap({
 layers:{ enableMouseHover:true }
 }); | **Property:** *enable*
 var maps = new ej.maps.Maps({
 layers:[{ highlightSettings:{ enable:true }]
 });

 maps.appendTo('#container')|

| Highlight Border Color | **Property:** *layers.shapeSettings.highlightStroke*
 \$("#container").ejMap({
 layers:{ shapeSettings:{ highlightStroke:'red' } }
 }); | **Property:** *layers.highlightSettings.border.color*
 var maps = new ej.maps.Maps({
 layers:[{ highlightSettings:{ border:{ color: 'green' } }]
 });

 maps.appendTo('#container')|

| Highlight Border Width | **Property:** *layers.shapeSettings.highlightBorderWidth*
 \$("#container").ejMap({
 layers:{ shapeSettings:{ highlightBorderWidth:'2' } }
 }); | **Property:** *layers.highlightSettings.border.width*
 var maps = new ej.maps.Maps({
 layers:[{ highlightSettings:{ border:{ width: 2 } }]
 });

 maps.appendTo('#container')|

| Highlight Opacity | Not Applicable | **Property:** *layers.layers.highlightSettings.opacity*
 var maps = new ej.maps.Maps({
 layers:[{ highlightSettings:{ opacity: 0.5 } }]
 });

 maps.appendTo('#container')|

| Selection Fill | **Property:** *layers.shapeSettings.selectionColor*
 \$("#container").ejMap({
 layers:{ shapeSettings:{ selectionColor:'blue' } }
 }); | **Property:** *layers.selectionSettings.fill*
 var maps = new ej.maps.Maps({
 layers:[{ selectionSettings:{ fill:'#D2691E' } }]
 });

 maps.appendTo('#container')|

| Selection Enable | **Property:** *layers.enableSelection*
 \$("#container").ejMap({
 layers:{ enableSelection:true }
 }); | **Property:** *layers.selectionSettings.enable*
 var maps = new ej.maps.Maps({
 layers:[{ selectionSettings:{ enable:true } }]
 });

 maps.appendTo('#container')|

| Selection Border Width | **Property:** *layers.selectionSettings.selectionStrokeWidth*
 \$("#container").ejMap({
 layers:{ selectionSettings:{ selectionStrokeWidth:'2' } }
 }); | **Property:** *layers.selectionSettings.border.width*
 var maps = new ej.maps.Maps({
 layers:[{ selectionSettings:{ border:{ width:2 } }]
 });

 maps.appendTo('#container')|

| Selection Border Color | **Property:** *layers.selectionSettings.selectionStroke*
 \$("#container").ejMap({
 layers:{ layers:[{ selectionSettings:{ selectionStroke:'red' } }]
 }); | **Property:** *layers.selectionSettings.border.color*
 var maps = new ej.maps.Maps({
 layers:[{ selectionSettings:{ border:{ color:'blue' } }]
 });

 maps.appendTo('#container')|

| Selection Opacity | Not Applicable | **Property:** *layers.selectionSettings.opacity*

 var maps = new ej.maps.Maps({
 layers:[{selectionSettings:{ opacity:2 }}]
 });
 maps.appendTo('#container') |

Navigation Line Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Visible | Not Applicable | **Property:** *layers.navigationLineSettings.visible*

 var maps = new ej.maps.Maps({
 layers:[{ navigationLineSettings:{ visible:true }}]
 });
 maps.appendTo('#container') |

| Width | Not Applicable | **Property:** *layers.navigationLineSettings.width*

 var maps = new ej.maps.Maps({
 layers:[{ navigationLineSettings:{ width:2 }}]
 });
 maps.appendTo('#container') |

| Longitude | Not Applicable | **Property:** *layers.navigationLineSettings.longitude*

 var maps = new ej.maps.Maps({
 layers:[{ navigationLineSettings:{ longitude: [-97.8717041015625, -89.6649169921875] }}]
 });
 maps.appendTo('#container') |

| Latitude | Not Applicable | **Property:** *layers.navigationLineSettings.latitude*

 var maps = new ej.maps.Maps({
 layers:[{ navigationLineSettings:{ latitude: [22.403410892712124, 21.282336521195344] }}]
 });
 maps.appendTo('#container') |

| DashArray | Not Applicable | **Property:** *layers.navigationLineSettings.dashArray*

 var maps = new ej.maps.Maps({
 layers:[{ navigationLineSettings:{ dashArray:"1,2" }}]
 });
 maps.appendTo('#container') |

| Color | Not Applicable | **Property:** *layers.navigationLineSettings.color*

 var maps = new ej.maps.Maps({
 layers:[{ navigationLineSettings:{ color:"green" }}]
 });
 maps.appendTo('#container') |

| Angle | Not Applicable | **Property:** *layers.navigationLineSettings.angle*

 var maps = new ej.maps.Maps({
 layers:[{ navigationLineSettings:{ angle:180 }}]
 });
 maps.appendTo('#container') |

| Arrow Position | Not Applicable | **Property:** *layers.navigationLineSettings.arrow.position*

 var maps = new ej.maps.Maps({
 layers:[{ navigationLineSettings:{ arrow:{ position:"start" }}]
 });
 maps.appendTo('#container') |

| Show Arrow | Not Applicable | **Property:** *layers.navigationLineSettings.arrow.showArrow*

 var maps = new ej.maps.Maps({
 layers:[{ navigationLineSettings:{ arrow:{ showArrow:true }}]
 });
 maps.appendTo('#container') |

| Arrow size | Not Applicable | **Property:** *layers.navigationLineSettings.arrow.size*

 var maps = new ej.maps.Maps({
 layers:[{ navigationLineSettings:{ arrow:{ size:2 }}]
 });
 maps.appendTo('#container') |

| Arrow Color | Not Applicable | **Property:** *layers.navigationLineSettings.arrow.color*

 var maps = new ej.maps.Maps({
 layers:[{ navigationLineSettings:{ arrow:{ color:'red' }}]
 });
 maps.appendTo('#container') |

| Arrow Offset | Not Applicable | **Property:** *layers.navigationLineSettings.arrow.offSet*
 var maps = new ej.maps.Maps({ layers: [{ navigationLineSettings: { arrow: { offSet: 10 } } }] });
 maps.appendTo('#container') |

Tooltip Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Tooltip Enable | **Property:** *layers.showTooltip*
 \$("#container").ejMap({ layers: { layers: { showTooltip: true } } }); | **Property:** *layers.tooltipSettings.visible*
 var maps = new ej.maps.Maps({ layers: { tooltipSettings: { visible: true } } });
 maps.appendTo('#container') |

| Tooltip Template | **Property:** *layers.tooltipTemplate*
 \$("#container").ejMap({ layers: { layers: { tooltipTemplate: "Template" } } }); | **Property:** *layers.tooltipSettings.visible*
 var maps = new ej.maps.Maps({ layers: { tooltipSettings: { visible: true } } });
 maps.appendTo('#container') |

| Value Path | Not Applicable | **Property:** *layers.tooltipSettings.valuePath*
 var maps = new ej.maps.Maps({ layers: { tooltipSettings: { valuePath: 'Population' } } });
 maps.appendTo('#container') |

| Format | Not Applicable | **Property:** *layers.tooltipSettings.format*
 var maps = new ej.maps.Maps({ layers: { tooltipSettings: { format: '\${State}
\${Population}' } } });
 maps.appendTo('#container') |

| Border Color | Not Applicable | **Property:** *layers.tooltipSettings.border.color*
 var maps = new ej.maps.Maps({ layers: { tooltipSettings: { border: { color: 'red' } } } });
 maps.appendTo('#container') |

| Border Width | Not Applicable | **Property:** *layers.tooltipSettings.border.width*
 var maps = new ej.maps.Maps({ layers: { tooltipSettings: { border: { width: " " } } } });
 maps.appendTo('#container') |

| Text Style | Not Applicable | **Property:** *layers.tooltipSettings.textStyle*
 var maps = new ej.maps.Maps({ layers: { tooltipSettings: { textStyle: { size: '15px', color: 'red', fontFamily: 'arial', fontWeight: 'bold', fontStyle: 'normal', opacity: 0.8 } } } });
 maps.appendTo('#container') |

Annotation Cutomization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Content | Not Applicable | **Property:** *legendSettings.annotations.content*
 var maps = new ej.maps.Maps({ annotations: { content: '<div>USA Population 2018</div>' } });
 maps.appendTo('#container') |

| Location X | Not Applicable | **Property:** *legendSettings.annotations.x*
 var maps = new ej.maps.Maps({ annotations: { x: '250px' } });
 maps.appendTo('#container') |

| Location Y | Not Applicable | **Property:** *legendSettings.annotations.y*
 var maps = new ej.maps.Maps({ annotations: { y: '150px' } });
 maps.appendTo('#container') |

| Vertical Alignment | Not Applicable | **Property:** *legendSettings.annotations.verticalAlignment*
legendSettings.annotations.verticalAlignment
 var maps = new ej.maps.Maps({
 annotations: { verticalAlignment:'Center' }
 }); maps.appendTo('#container') |

| Horizontal Alignment | Not Applicable | **Property:** *legendSettings.annotations.horizontalAlignment*
legendSettings.annotations.horizontalAlignment
 var maps = new ej.maps.Maps({
 annotations: { horizontalAlignment:'Center' }
 }); maps.appendTo('#container') |

| Zindex | Not Applicable | **Property:** *legendSettings.annotations.zIndex*
legendSettings.annotations.zIndex
 var maps = new ej.maps.Maps({
 annotations: { zIndex:'-1' }
 }); maps.appendTo('#container') |

Maps Other Properties Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Projection Type | Not Applicable | **Property:** *projectionType*
projectionType
 var maps = new ej.maps.Maps({
 projectionType: 'Mercator'
 }); maps.appendTo('#container') |

| Background | **Property:** *background*
background
 \$("#container").ejMap({
 background:'red'
 }); | **Property:** *background*
background
 var maps = new ej.maps.Maps({
 background: 'red'
 }); maps.appendTo('#container') |

| Enable Group Separator | **Property:** *enableGroupSeparator*
enableGroupSeparator
 \$("#container").ejMap({
 enableGroupSeparator:true
 }); | **Property:** *useGroupingSeparator*
useGroupingSeparator
 var maps = new ej.maps.Maps({
 useGroupingSeparator: true
 }); maps.appendTo('#container') |

| Base Layer Index | **Property:** *baseMapIndex*
baseMapIndex
 \$("#container").ejMap({
 baseMapIndex:0
 }); | **Property:** *baseLayerIndex*
baseLayerIndex
 var maps = new ej.maps.Maps({
 baseLayerIndex: 0
 }); maps.appendTo('#container') |

| locale | **Property:** *locale*
locale
 \$("#container").ejMap({
 locale:'en-us'
 }); | Not Applicable |

| Responsive | **Property:** *isResponsive*
isResponsive
 \$("#container").ejMap({
 isResponsive:true
 }); | Not Applicable |

| Enable Pan | **Property:** *enablePan*
enablePan
 \$("#container").ejMap({
 enablePan:true
 }); | Not Applicable |

| Enable Navigation | **Property:** *navigationControl.enableNavigation*
navigationControl.enableNavigation
 \$("#container").ejMap({
 navigationControl:{ enableNavigation:true }
 }); | Not Applicable |

| Navigation Orientation | **Property:** *navigationControl.orientation*
navigationControl.orientation
 \$("#container").ejMap({
 navigationControl:{ orientation:'vertical' }
 }); | Not Applicable |

| Navigation Dock Position | **Property:** *navigationControl.dockPosition*
navigationControl.dockPosition
 \$("#container").ejMap({
 navigationControl:{ dockPosition:'centerleft' }
 }); | Not Applicable |

| Navigation Absolute Position | **Property:** *navigationControl.absolutePosition*
navigationControl.absolutePosition
 \$("#container").ejMap({
 navigationControl:{ absolutePosition:{ x: 100, y : 100 } }
 }); | Not Applicable |

| Dragging Selection | **Property:** *draggingOnSelection*
draggingOnSelection
 \$("#container").ejMap({
 draggingOnSelection : true
 }); | Not Applicable |

| Resize | **Property:** *enableResize*
 \$("#container").ejMap({ enableResize : true
 }); | Not Applicable |

| Enable Animation | **Property:** *enableAnimation*
 \$("#container").ejMap({ enableAnimation : true
 }); | Not Applicable |

| Enable Layer Animation | **Property:** *enableLayerChangeAnimation*
 \$("#container").ejMap({ enableLayerChangeAnimation : true
 }); | Not Applicable |

| Center Position | **Property:** *centerPosition*
 \$("#container").ejMap({ centerPosition:[90.52734374999999,30.41078179084589]
 }); | **Property:** *centerPosition*
 var maps = new ej.maps.Maps({ centerPosition:{ latitude:
 30.41078179084589,longitude: 90.52734374999999 }
 }); maps.appendTo('#container') |

Events

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Shape Selected | **Property:** *shapeSelected*
 \$("#container").ejMap({ shapeSelected:'MapShapeSelected'
 }); | **Property:** *shapeSelected*
 var maps = new ej.maps.Maps({ shapeSelected:'MapShapeSelected'
 }); maps.appendTo('#container') |

| Marker Selected | **Property:** *markerSelected*
 \$("#container").ejMap({ markerSelected:'MapMarkerSelected'
 }); | **Property:** *markerClick*
 var maps = new ej.maps.Maps({ markerClick:'MapMarkerSelected'
 }); maps.appendTo('#container') |

| Marker Move | **Property:** *markerEnter*
 \$("#container").ejMap({ markerEnter:'MapMarkerMove'
 }); | **Property:** *markerMouseMove*
 var maps = new ej.maps.Maps({ markerMouseMove:'MapMarkerMove'
 }); maps.appendTo('#container') |

| Marker Leave | **Property:** *markerLeave*
 \$("#container").ejMap({ markerLeave:'MapMarkerLeave'
 }); | Not Applicable |

| Legend Item Rendering | **Property:** *legendItemRendering*
 \$("#container").ejMap({ legendItemRendering:'MapLegendItemRendering'
 }); | Not Applicable |

| Display Text Rendering | **Property:** *displayTextRendering*
 \$("#container").ejMap({ displayTextRendering:'MapDisplayTextRendering'
 }); | **Property:** *dataLabelRendering*
 var maps = new ej.maps.Maps({ dataLabelRendering:'MapDataLabelRendering'
 }); maps.appendTo('#container') |

| Legend Item Click | **Property:** *legendItemClick*
 \$("#container").ejMap({ legendItemClick:'MapLegendItemClick'
 }); | Not Applicable |

| Bubble Rendering | **Property:** *bubbleRendering*
 \$("#container").ejMap({ bubbleRendering:'MapBubbleRendering'
 }); | **Property:** *bubbleRendering*
 var maps = new ej.maps.Maps({ bubbleRendering:'MapBubbleRendering'
 }); maps.appendTo('#container') |

| Shape Rendering | **Property:** *shapeRendering*
 \$("#container").ejMap({ shapeRendering:'MapShapeRendering'
 }); | **Property:** *shapeRendering*
 var maps = new

```

ej.maps.Maps({ <br/> &#160; shapeRendering:'MapShapeRendering' <br/> }); <br/>
maps.appendTo('#container') |

| Zoomed In | Property: zoomedIn<br/><br/> $("#container").ejMap({ <br/> &#160;
zoomedIn:'MapZooming' <br/> }); | Not Applicable |

| Render Completed | Property: onRenderComplete<br/><br/> $("#container").ejMap({ <br/> &#160;
onRenderComplete:'MapRenderCompleted' <br/> }); | Property: loaded<br/><br/> var maps = new
ej.maps.Maps({ <br/> &#160; loaded:'MapRenderCompleted' <br/> }); <br/>
maps.appendTo('#container') |

| Panned | Property: panned<br/><br/> $("#container").ejMap({ <br/> &#160; panned:'MapPanned'
<br/> }); | Not Applicable |

| zoomed Out | Property: zoomedOut<br/><br/> $("#container").ejMap({ <br/> &#160;
zoomedOut:'MapZoomedOut' <br/> }); | Not Applicable |

| Mouse Over | Property: mouseover<br/><br/> $("#container").ejMap({ <br/> &#160;
mouseover:'MapMouseOver' <br/> }); | Not Applicable |

| Mouse Leave | Property: mouseleave<br/><br/> $("#container").ejMap({ <br/> &#160;
mouseover:'MapMouseLeave' <br/> }); | Not Applicable |

| Click | Property: click<br/><br/> $("#container").ejMap({ <br/> &#160; click:'ClickOnMap' <br/> }); |
Property: click<br/><br/> var maps = new ej.maps.Maps({ <br/> &#160; click:'ClickOnMap' <br/> });
<br/> maps.appendTo('#container') |

| Double Click | Property: doubleClick<br/><br/> $("#container").ejMap({ <br/> &#160;
doubleClick:'DoubleClickOnMap' <br/> }); | Property: doubleClick<br/><br/> var maps = new
ej.maps.Maps({ <br/> &#160; doubleClick:'DoubleClickOnMap' <br/> }); <br/>
maps.appendTo('#container') |

| Right Click | Property: rightClick<br/><br/> $("#container").ejMap({ <br/> &#160;
rightClick:'RightClickOnMap' <br/> }); | Property: rightClick<br/><br/> var maps = new ej.maps.Maps({
<br/> &#160; rightClick:'RightClickOnMap' <br/> }); <br/> maps.appendTo('#container') |

| Initial Load | Property: onLoad<br/><br/> $("#container").ejMap({ <br/> &#160; onLoad:'loadOnMap'
<br/> }); | Property: load<br/><br/> var maps = new ej.maps.Maps({ <br/> &#160; load:'loadOnMap'
<br/> }); <br/> maps.appendTo('#container') |

| Before Print | Not Applicable | Property: beforePrint<br/><br/> var maps = new ej.maps.Maps({ <br/>
&#160; beforePrint:'MapBeforePrint' <br/> }); <br/> maps.appendTo('#container') |

| Resize | Not Applicable | Property: resize<br/><br/> var maps = new ej.maps.Maps({ <br/> &#160;
resize:'ResizeOnMap' <br/> }); <br/> maps.appendTo('#container') |

| Tooltip Render | Not Applicable | Property: tooltipRender<br/><br/> var maps = new ej.maps.Maps({
<br/> &#160; tooltipRender:'MapTooltipRender' <br/> }); <br/> maps.appendTo('#container') |

| Item Selection | Not Applicable | Property: itemSelection<br/><br/> var maps = new ej.maps.Maps({
<br/> &#160; itemSelection:'MapItemSelection' <br/> }); <br/> maps.appendTo('#container') |

| Item Highlight | Not Applicable | Property: itemHighlight<br/><br/> var maps = new ej.maps.Maps({
<br/> &#160; itemHighlight:'MapItemHighlight' <br/> }); <br/> maps.appendTo('#container') |

```

| Shape Highlight | Not Applicable | **Property:** *shapeHighlight*

 var maps = new ej.maps.Maps({
 shapeHighlight:'MapShapeHighlight'
 });
 maps.appendTo('#container') |

| Layer Rendering | Not Applicable | **Property:** *layerRendering*

 var maps = new ej.maps.Maps({
 layerRendering:'MapLayerRendering'
 });
 maps.appendTo('#container') |

| Marker Rendering | Not Applicable | **Property:** *markerRendering*

 var maps = new ej.maps.Maps({
 markerRendering:'MapMarkerRendering'
 });
 maps.appendTo('#container') |

| Bubble Mouse Move | Not Applicable | **Property:** *bubbleMouseMove*

 var maps = new ej.maps.Maps({
 bubbleMouseMove:'MouseMoveOnBubble'
 });
 maps.appendTo('#container') |

| Bubble Mouse Move | Not Applicable | **Property:** *annotationRendering*

 var maps = new ej.maps.Maps({
 annotationRendering:'MapAnnotationRendering'
 });
 maps.appendTo('#container') |

| Animation Complete | Not Applicable | **Property:** *animationComplete*

 var maps = new ej.maps.Maps({
 animationComplete:'MapAnimationComplete'
 });
 maps.appendTo('#container') |

How To

Annotation in EJ2 JavaScript Maps control

Annotations are used to mark the specific area of interest in the Maps with texts, shapes, or images. Any number of annotations can be added to the Maps component.

Initialize the Maps control with annotation option, text content or ID of an HTML element or an HTML string can be specified to render a new element that needs to be displayed in the Maps by using the [content](#) property. To specify the content position with [x](#) and [y](#) properties as mentioned in the following example.

INDEX.TS

```
import { Africa_Continent } from './Africa_Continent.ts';
import { Maps, Annotations } from '@syncfusion/ej2-maps';
Maps.Inject(Annotations);
// initialize Maps component
let map: Maps = new Maps({
  layers: [
    {
      shapeData: Africa_Continent,
      shapeSettings: {
        fill: 'url(#grad1)'
      }
    }
  ],
  annotations: [
    {
      content: '#maps-annotation', // To insert the text content
      x: '0%', y: '70%'
    },
    {
```



```

        content: '#compass-maps', // To insert the image
        x: '80%', y: '5%'
    }
}
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="africa_continent.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="container" style="height: 500px; width: 700px">
        <div id="element"></div>
    </div>
    <svg height="150" width="400">
        <defs>
            <linearGradient id="grad1" x1="0%" y1="0%" x2="0%" y2="100%">
                <stop offset="0%" style="stop-color:#C5494B;stop-
opacity:1"></stop>
                <stop offset="100%" style="stop-color:#4C134F;stop-
opacity:1"></stop>
            </linearGradient>
        </defs>
    </svg>
    <!-- annotation content -->
    <div id="maps-annotation" style="display: none;">
        <div id="annotation">
            <div>
                <p style="margin-left:10px;font-size:13px;font-
weight:500">Facts about Africa</p>
            </div>
            <hr style="margin-top:-3px;margin-bottom:10px;border:0.5px solid
#DDDDDD">
            <div>
                <ul style="list-style-type:disc; margin-left:-20px;margin-
bottom:2px; font-weight:400">
                    <li>Africa is the second largest and second most
populated continent in the world.</li>
                    <li style="padding-top:5px;">Africa has 54 sovereign
states and 10 non-sovereign territories.</li>
                    <li style="padding-top:5px;">Algeria is the largest
country in Africa, where as Mayotte is the
smallest.</li>

```

```

        </ul>
      </div>
    </div>
  </div>
  <div id="compass-maps" style="display: none;">
    
    </div>
    <style>
      #annotation {
        color: #DDDDDD;
        font-size: 12px;
        font-family: Roboto;
        background: #3E464C;
        margin: 20px;
        padding: 10px;
        -webkit-border-radius: 2px;
        -moz-border-radius: 2px;
        border-radius: 2px;
        width: 300px;
        -moz-box-shadow: 0px 2px 5px #666;
        -webkit-box-shadow: 0px 2px 5px #666;
        box-shadow: 0px 2px 5px #666;
      }
      .country-label {
        color: white;
        font-size: 25px;
      }
    </style>
    <script>
      var ele = document.getElementById('container');
      if (ele) {
        ele.style.visibility = "visible";
      }
    </script>
    <script src="index.js" type="text/javascript"></script>
  </body>
</html>

```

```

<svg height="150" width="400">
<defs>
<linearGradient id="grad1" x1="0%" y1="0%" x2="0%" y2="100%">
<stop offset="0%" style="stop-color:#C5494B;stop-opacity:1"></stop>
<stop offset="100%" style="stop-color:#4C134F;stop-opacity:1"></stop>
</linearGradient>
</defs>
</svg>

```

```
<div id="maps-annotation" style="display: none;">
<div id="annotation">
<div>
<p style="margin-left:10px;font-size:13px;font-weight:500">Facts about Africa</p>
</div>
<hr style="margin-top:-3px;margin-bottom:10px;border:0.5px solid #DDDDDD">
<div>
<ul style="list-style-type:disc; margin-left:-20px;margin-bottom:2px; font-weight:400">
<li>Africa is the second largest and second most populated continent in the world.</li>
<li style="padding-top:5px;">Africa has 54 sovereign states and 10 non-sovereign territories.
</li>
<li style="padding-top:5px;">Algeria is the largest country in Africa, where as Mayotte is the
smallest.</li>
</ul>
</div>
</div>
</div>
<div id="compass-maps" style="display: none;">

</div>
<style>
annotation {
color: #DDDDDD;
font-size: 12px;
font-family: Roboto;
background: #3E464C;
margin: 20px;
padding: 10px;
border-radius: 2px;
width: 300px;
box-shadow: 0px 2px 5px #666;
}
</style>
`
```

Custom path in EJ2 JavaScript Maps control

Maps control can be customized as the desired layout using the custom path map feature. Here, the Maps control has been showcased with normal geometry type shapes to represent the bus seat selection layout. Please refer to the following example to render the bus seat selection.

<!-- markdownlint-disable MD031 -->

INDEX.TS

```
import { seat } from './seat.ts';
import { Maps, Selection } from '@syncfusion/ej2-maps';
Maps.Inject(Selection);
// initialize Maps component
let map: Maps = new Maps({
    layers: [
        {
            geometryType: 'Normal',
            shapeData: seat,
            selectionSettings: {
                enable: true,
                opacity: 1,
                enableMultiSelect: true
            }
        }
    ],
    height: '400'
}, '#container');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="seat.js"></script>

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div class="col-lg-9 control-section">
    <div style="width:200px;margin:auto;padding-bottom:20px">
      <div style="padding-left:30px;font-size:20px;font-
weight:400;">Bus seat selection</div>
    </div>
    <div style="border: 3px solid
darkgray;width:200px;display:block;margin:auto;border-radius:5px">
      <div id="container"></div>
```

```

        </div>
    </div>

    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

```

<div class="col-lg-9 control-section">
<div style="width:200px;margin:auto;padding-bottom:20px">

<div style="padding-left:30px;font-size:20px;font-weight:400;">Bus seat selection</div>
</div>

<div style="border: 3px solid darkgray;width:200px;display:block;margin:auto;border-radius:5px">

<div id="maps"></div>
</div>
</div>

```

Drilldown in EJ2 JavaScript Maps control

By clicking a continent, all the countries available in that continent can be viewed using the drill-down feature. For example, the countries in the **Africa** continent have been showcased here. To showcase all the countries in **Africa** continent by clicking the [shapeSelected](#) event as mentioned in the following example.

<!-- markdownlint-disable MD031 -->

INDEX.TS

```

import { world_map } from './world-map.ts';
import { Africa_Continent } from './Africa_Continent.ts';
import { default_data } from './data.ts';
import { Maps, Highlight, IShapeSelectedEventArgs, shapeSelected, Marker } from
'@syncfusion/ej2-maps';
Maps.Inject(Highlight, Marker );
export interface ShapeData {
    continent?: string;
}
// Initialize Maps component.
let map: Maps = new Maps({

```

```

shapeSelected: (args: IShapeSelectedEventArgs): void => {
    let shape: string = (args.shapeData as ShapeData).continent;
    if (shape === 'Africa') {
        map.baseLayerIndex = 1;
        map.refresh();
    }
},
layers: [
    {
        layerType: 'Geometry',
        shapeData: world_map,
        shapePropertyPath: 'continent',
        shapeDataPath: 'continent',
        dataSource: default_data,
        shapeSettings: {
            colorValuePath: 'drillColor'
        },
        markerSettings: [{
            visible: true,
            template: '<div id="marker3" class="markerTemplate">Africa'
+
                '</div>',
            dataSource: [
                { latitude: 10.97274101999902, longitude: 16.390625 }
            ],
            animationDuration: 0
        }]
    },
    {
        layerType: 'Geometry',
        shapeData: Africa_Continent,
        shapeSettings: {
            fill: '#80306A'
        },
        highlightSettings: {
            enable: true,
            fill: '#80306A'
        }
    }
]
});
map.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <script src="world-map.js"></script>
    <script src="africa.js"></script>

```

```

    <script src="data.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

```

,
<div id="mapdrilldown"></div>
<style>
.markerTemplate {
font-size: 12px;
color: white;
text-shadow: 0px 1px 1px black;
font-weight: 500
}
.markerTemplate {
height: 30px;
width: 30px;
display: block;
margin: auto;
}
</style>
,

```

Marker type in EJ2 JavaScript Maps control

Add different types of markers

Different marker objects can be added to the Maps component using the marker settings. To update different marker settings in Maps, please follow the given steps:

```
<!-- markdownlint-disable MD034 -->
```

Step 1:

Initialize the Maps control with marker settings. Here, a marker has been added with specified latitude and longitude of California by using the [dataSource](#) property. To customize the shape of the marker using the [shape](#) property and change the border color and width of the marker using the [border](#) property as mentioned in the following example.

INDEX.TS

```
import { world_map } from './world-map.ts';
import { Maps, Marker, MarkerSettings } from '@syncfusion/ej2-maps';
// Initialize Map component.
let map: Maps = new Maps({
  // Initializing Map with Marker settings.
  layers: [
    {
      shapeData: world_map,
      markerSettings: [
        {
          dataSource: [
            { latitude: 40.7424509, longitude: -74.0081468, city:
'New York' }
          ],
          visible: true,
          shape: 'Circle',
          fill: 'white',
          width: 3,
          animationDuration: 0,
          border: {width: 2, color: '#333'}
        }
      ]
    }
  ]
});
map.appendTo('#element'); // render initialized Map.
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
```



```

</div>
<script>
  var ele = document.getElementById('container');
  if (ele) {
    ele.style.visibility = "visible";
  }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Step 2:

Customize the above option for n number of markers as mentioned in the following example.

INDEX.TS

```

import { world_map } from './world-map.ts';
import { Maps, Marker, MarkerSettings } from '@syncfusion/ej2-maps';
Maps.Inject(Marker);
// initialize Maps component
let map: Maps = new Maps({
  layers: [
    {
      shapeData: world_map,
      markerSettings: [
        {
          dataSource: [
            { latitude: 37.6276571, longitude: -122.4276688,
city: 'San Bruno' },
          ],
          visible: true,
          shape: 'Circle',
          fill: 'white',
          width: 3,
          animationDuration: 0,
          border: { width: 2, color: '#333' }
        },
        {
          dataSource: [
            { latitude: 33.5302186, longitude: -117.7418381,
city: 'Laguna Niguel' },
          ],
          visible: true,
          shape: 'Rectangle',
          fill: 'yellow',
          width: 15,
          height: 4,
          animationDuration: 0,
          border: { width: 2, color: '#333' }
        },
        {
          dataSource: [
            { latitude: 40.7424509, longitude: -74.0081468,
city: 'New York' }
          ],
          visible: true,

```

```

        shape: 'Diamond',
        fill: 'white',
        width: 10,
        height: 10,
        animationDuration: 0,
        border: {width: 2, color: 'blue'}
      }
    ]
  }
}
});
map.appendTo('#element'); // render initialized Map.

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="world-map.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Multiple layer in EJ2 JavaScript Maps control

The multilayer support allows loading multiple shape files in a single container and enables Maps to display more information. The shape layer is the main layer of the Maps. Multiple layers can be added in a shape layer as **SubLayer** using the [type](#) property.

INDEX.TS

```

import { usa_map } from './usa.ts';
import { california } from './california.ts';
import { texas } from './texas.ts';
import { Maps } from '@syncfusion/ej2-maps';

```

```
// initialize Maps component
let map: Maps = new Maps({
  layers: [
    {
      shapeData: usa_map,
      shapeSettings: {
        fill: '#E5E5E5',
        border: {
          color: 'black',
          width: 0.1
        }
      }
    },
    {
      shapeData: texas,
      type: 'SubLayer',
      shapeSettings: {
        fill: 'rgba(141, 206, 255, 0.6)',
        border: {
          color: '#1a9cff',
          width: 0.25
        }
      }
    },
    {
      shapeData: california,
      type: 'SubLayer',
      shapeSettings: {
        fill: 'rgba(141, 206, 255, 0.6)',
        border: {
          color: '#1a9cff',
          width: 0.25
        }
      }
    }
  ]
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <script src="usa.js"></script>
  <script src="california.js"></script>
  <script src="texas.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>
  <div id="container" style="height: 500px; width: 700px">
    <div id="element"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Navigation line in EJ2 JavaScript Maps control

To navigate a particular country by setting the center position zooming using the [centerPosition](#) and [zoomFactor](#) property as demonstrated in the following sample. The center position is used to configure the zoom level of maps, and zoom factor is used to specify the center position where the map should be displayed.

To navigate to a particular country, follow the given steps:

Step 1:

Initialize the maps and add country list for drop-down elements in the load event.

INDEX.TS

```

import { Maps, ILoadEventArgs, Zoom } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
import { latLongPoints } from './latLongPoints.ts';
Maps.Inject(Zoom);
// initialize Maps component
let map: Maps = new Maps(
  {
    zoomSettings: {
      enable: true,
      zoomOnClick: true
    },
    layers: [
      {
        animationDuration: 1000,
        shapeData: world_map,
        dataSource: latLongPoints,
        shapePropertyPath: 'name',
        shapeDataPath: 'name',
        selectionSettings: {
          enable: true,
          fill: 'red'
        },
        shapeSettings: {
          autofill: true,
          palette: [
            '#4A3825', '#736F3D', '#F2DABD', '#BF9D7E',
            '#7F6039', '#7F715F', '#70845D', '#CC995C', '#736F3D', '#89541B'
          ]
        }
      }
    ]
  }
);

```

```

    }
    ],
    load: function (args: ILoadEventArgs) {
        let element: HTMLSelectElement =
<HTMLSelectElement>document.getElementById("countriesCombo");
        for (let i: number = 0; i < latLongPoints.length; i++) {
            let item = latLongPoints[i];
            let selectBoxOption = document.createElement("option");
            selectBoxOption.value = item.name;
            selectBoxOption.text = item.name;
            element.add(selectBoxOption, null);
        }
    }
}
);
// render initialized Map
map.appendTo('#container');
document.getElementById("countriesCombo").onchange = function () {
    let lat =
latLongPoints[(<HTMLSelectElement>this).selectedIndex].latitude;
    let long =
latLongPoints[(<HTMLSelectElement>this).selectedIndex].longitude;
    map.centerPosition.latitude = lat;
    map.centerPosition.longitude = long;
    map.layers[0].animationDuration = 1000;
    map.refresh();
    let group =
document.getElementById("container_LayerIndex_0_Polygon_Group");
    for (let j: number = 0; j < group.children.length; j++) {
        if (group.children[j].getAttribute("aria-label") ===
(<HTMLSelectElement>this).value) {
            let element = group.children[j];
            element.setAttribute("fill", "red");
            break;
        }
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="world-map.js"></script>
    <script src="latLongPoints.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>

```

```

<body>

  <div>
    <select size="1" style="margin: 10px;width: 180px;margin-top: 15px;
border-width: 1px;border-color: lightgray;margin-left: 15px;height: 30px"
id="countriesCombo">
    </select>
  </div>
  <div id="container">
  </div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Step 2:

Set the center position latitude and longitude as latitude and longitude of selected country by drop-downs, and then set the `zoomFactor` to navigate to the particular selected country.

To refresh the maps, invoke the `refresh` method.

INDEX.TS

```

import { Maps, ILoadEventArgs, Zoom } from '@syncfusion/ej2-maps';
import { world_map } from './world-map.ts';
import { latLongPoints } from './latLongPoints.ts';
Maps.Inject(Zoom);
// initialize Maps component
let map: Maps = new Maps(
  {
    zoomSettings: {
      enable: true,
      zoomOnClick: true
    },
    layers: [
      {
        animationDuration: 1000,
        shapeData: world_map,
        dataSource: latLongPoints,
        shapePropertyPath: 'name',
        shapeDataPath: 'name',
        selectionSettings: {
          enable: true,
          fill: 'red'
        },
        shapeSettings: {
          autofill: true,
          palette: ['#4A3825', '#736F3D', '#F2DABD', '#BF9D7E',
            '#7F6039', '#7F715F', '#70845D', '#CC995C', '#736F3D', '#89541B']
        },
      }
    ]
  }
);

```

```

    ],
    load: function (args: ILoadEventArgs) {
        let element: HTMLSelectElement =
<HTMLSelectElement>document.getElementById("countriesCombo");
        for (let i: number = 0; i < latLongPoints.length; i++) {
            let item = latLongPoints[i];
            let selectBoxOption = document.createElement("option");
            selectBoxOption.value = item.name;
            selectBoxOption.text = item.name;
            element.add(selectBoxOption, null);
        }
    }
};
// render initialized Map
map.appendTo('#container');
document.getElementById("countriesCombo").onchange = function () {
    let lat =
latLongPoints[(<HTMLSelectElement>this).selectedIndex].latitude;
    let long =
latLongPoints[(<HTMLSelectElement>this).selectedIndex].longitude;
    map.centerPosition.latitude = lat;
    map.centerPosition.longitude = long;
    map.zoomSettings.zoomFactor = 5;
    map.layers[0].animationDuration = 1000;
    map.refresh();
    let group =
document.getElementById("container_LayerIndex_0_Polygon_Group");
    for (let j: number = 0; j < group.children.length; j++) {
        if (group.children[j].getAttribute("aria-label") ===
(<HTMLSelectElement>this).value) {
            let element = group.children[j];
            element.setAttribute("fill", "red");
            break;
        }
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Maps</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="world-map.js"></script>
    <script src="latLongPoints.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>

```

```

<body>

  <div>
    <select size="1" style="margin: 10px;width: 180px;margin-top: 15px;
border-width: 1px;border-color: lightgray;margin-left: 15px;height: 30px"
id="countriesCombo">
    </select>
  </div>
  <div id="container">
  </div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Zooming in EJ2 JavaScript Maps control

The center position zooming can be achieved by using the [centerPosition](#) and [zoomFactor](#) properties as mentioned in the following example. The center position is used to configure the zoom level of Maps, and the zoom factor is used to specify the center position where the Maps should be displayed.

INDEX.TS

```

import { world_map } from './world-map.ts';
import { Maps, Zoom, ZoomSettings } from '@syncfusion/ej2-maps';
Maps.Inject(Zoom);
// initialize Maps component
let map: Maps = new Maps({
  zoomSettings: {
    enable: true,
    zoomFactor: 13
  },
  centerPosition: {
    latitude: 25.54244147012483,
    longitude: -89.62646484375
  },
  layers: [
    {
      shapeData: world_map
    }
  ]
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 Maps</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```



```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<script src="world-map.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container" style="height: 500px; width: 700px">
  <div id="element"></div>
</div>
<script>
  var ele = document.getElementById('container');
  if (ele) {
    ele.style.visibility = "visible";
  }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

MaskedTextBox

Mask configuration in EJ2 JavaScript Maskedtextbox control

The mask is a combination of standard and custom mask elements that validates the user input based on its behavior.

When the mask value is empty, the MaskedTextBox behaves as an input element with text type.

Standard mask elements

The following table shows the list of mask elements and its behavior based on [MSDN](#) standard.

The mask can be formed by combining any one or more of these mask elements.

Mask Element	Description
-----	-----
0	Digit required. This element will accept any single digit from 0 to 9.
9	Digit or space, optional.
#	Digit or space, optional, Plus(+) and minus(-) signs are allowed.
L	Letter required. It will accept letters a-z and A-Z.
?	Letter or space, optional.
&	Requires a character.
C	Character or space, optional.
A	Alphanumeric (A-Za-z0-9) required.
a	Alphanumeric (A-Za-z0-9) or space, optional.
<	Shift down. Converts all characters to lower case.

- | > | Shift up. Converts all characters to upper case. |
- | | | Disable a previous shift up or shift down. |
- | \\\ | Escapes a mask character, turning it into a literal. |
- | All other characters | Literals. All non-mask elements (literals) will appear as themselves within MaskedTextBox. |

The following example demonstrates the usage of standard mask elements.

INDEX.TS

```
import { MaskedTextBox } from '@syncfusion/ej2-inputs';
// initializes the MaskedTextBox component
let mask1: MaskedTextBox = new MaskedTextBox({
    // sets mask with the mask element '#' which accepts any single
    digit from '0' to '9',
    // space, + and - signs
    mask: '#####',
    placeholder: 'Mask ##### (ex: 012+-)',
    floatLabelType: 'Always'
});
mask1.appendTo('#mask1');
let mask2: MaskedTextBox = new MaskedTextBox({
    // sets mask format with the mask element 'L' which allows only
    alphabets('A-Z and a-z')
    mask: 'LLLLLL',
    placeholder: 'Mask LLLLLL (ex: Sample)',
    floatLabelType: 'Always'
});
mask2.appendTo('#mask2');
let mask3: MaskedTextBox = new MaskedTextBox({
    // sets mask format with the mask element '&' which allows
    `alphabets`, `numbers`
    // and `special characters`
    mask: '#####',
    placeholder: 'Mask ##### (ex: A12@#)',
    floatLabelType: 'Always'
});
mask3.appendTo('#mask3');
let mask4: MaskedTextBox = new MaskedTextBox({
    // sets mask format with the mask element `>` which converts all
    characters that follow
    // to upper case and `<` which converts all characters that follow
    to lower case
    mask: '>LLL<LLL',
    placeholder: 'Mask >LLL<LL (ex: SAMple)',
    floatLabelType: 'Always'
});
mask4.appendTo('#mask4');
let mask5: MaskedTextBox = new MaskedTextBox({
    // sets mask format with the mask element '\\' which turns mask
    element `A` into
    // a literal and it displays the alphabet `A`
    mask: '\\A999',
    placeholder: 'Mask \\A999 (ex: A321)',
    floatLabelType: 'Always'
});
```

```
});
mask5.appendTo('#mask5');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 MaskedTextBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript MaskedTextBox Component
With Standard Masks">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="mask1" type="text">
    </div>
    <div class="wrap">
      <input id="mask2" type="text">
    </div>
    <div class="wrap">
      <input id="mask3" type="text">
    </div>
    <div class="wrap">
      <input id="mask4" type="text">
    </div>
    <div class="wrap">
      <input id="mask5" type="text">
    </div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
```

```

        visibility: hidden;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue', 'calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .wrap {
        margin: 20px auto;
        width: 240px;
    }
}

```

Custom mask elements

Other than the above standard mask elements, the mask can be configured with the custom characters or regular expression to define a custom behavior.

Custom characters

You can define any of the non-mask element as the mask element and its behavior through the [customCharacters](#) property.

In the following example, non-mask element **P** accepts the values **P, A, p, a**, and **M** accepts the values **M, m** as mentioned in the custom characters collection.

INDEX.TS

```

import { MaskedTextBox } from '@syncfusion/ej2-inputs';
// initializes the MaskedTextBox component
let mask: MaskedTextBox = new MaskedTextBox({
    // sets custom characters collection for non-mask elements 'P' and 'M'
    customCharacters: {
        P: 'P,A,a,p',
        M: 'M,m'
    },
    // sets mask format to the MaskedTextBox
    mask: '00:00 >PM',
    placeholder: 'Time (ex: 10:00 PM, 10:00 AM)',
    floatLabelType: 'Always'
});
mask.appendTo('#mask');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 MaskedTextBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript MaskedTextBox Component">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="mask" type="text">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 240px;
}
.label {
    font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';
    font-size: 14px;
}

```

Regular expression

Instead of the mask element, you can define your own regular expression to validate the input of a particular input place. The regular expressions should be wrapped by the square brackets (e.g., [Regex]).

In the following example, regular expression has been set for each input places.

INDEX.TS

```
import { MaskedTextBox } from '@syncfusion/ej2-inputs';  
// Render the Masked Textbox  
let mask: MaskedTextBox = new MaskedTextBox({  
  placeholder: 'IP Address (ex: 212.212.111.222) ',  
  floatLabelType: 'Always',  
  mask: '[0-2][0-9][0-9].[0-2][0-9][0-9].[0-2][0-9][0-9].[0-2][0-9][0-9]',  
});  
mask.appendTo('#mask');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 MaskedTextBox</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="TypeScript MaskedTextBox Component">  
  <meta name="author" content="Syncfusion">  
  <link href="styles.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">  
  
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>  
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>  
</head>  
<body>  
  
  <div id="container">  
    <div class="wrap">  
      <input id="mask" type="text">  
    </div>  
  </div>  
<script>  
var ele = document.getElementById('container');  
if(ele) {  
  ele.style.visibility = "visible";  
}  
</script>  
<script src="index.js" type="text/javascript"></script>  
</body></html>
```

STYLES.CSS

```
#container {  
  visibility: hidden;  
}
```

```
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  margin: 0 auto;
  width: 240px;
}
```

Prompt character

The Prompt character is a prompting symbol in the MaskedTextBox for the mask elements. The symbol is used to show the input positions in the MaskedTextBox. You can customize the prompt character of MaskedTextBox by using the [promptChar](#) property.

The following example demonstrates the MaskedTextBox with customized prompt character as `*`.

INDEX.TS

```
import { MaskedTextBox } from '@syncfusion/ej2-inputs';
// initializes the MaskedTextBox component
let mask: MaskedTextBox = new MaskedTextBox({
  // sets the prompting symbol to the MaskedTextBox
  promptChar: "#",
  // sets mask format to the MaskedTextBox
  mask: '999-999-9999'
});
mask.appendTo('#mask');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 MaskedTextBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript MaskedTextBox Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```

<body>

  <div id="container">
    <div class="wrap">
      <input id="mask" type="text">
    </div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  margin: 0 auto;
  width: 240px;
}
.label {
  font-family: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif';
  font-size: 14px;
}

```

Accessibility in EJ2 JavaScript Maskedtextbox control

The Maskedtextbox component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Maskedtextbox component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The
component does not meet the requirement.</div>
```

WAI-ARIA attributes

The MaskedTextBox is characterized with complete ARIA Accessibility support that helps to access through the on-screen readers and other assistive technology devices. This component is designed with the reference of the guidelines document given in [WAI ARAI Accessibility practices](#).

The MaskedTextBox uses the `textbox` role and following ARIA properties for its element based on its state.

| Property | Functionality |

| --- | --- |

| aria-live | The `aria-live` attribute indicates the priority of updates to a live region. |

| aria-disabled | The `aria-disabled` property indicates the disabled state of the MaskedTextBox. |

| aria-valuenow | The `aria-valuenow` property specifies the current value of the MaskedTextBox. |

| aria-invalid | The `aria-invalid` property indicates that the user input is incorrect or not within the acceptable ranges. |

| aria-placeholder | The `aria-placeholder` is a short hint to help the users with data entry when the MaskedTextBox has no value. |

| aria-labelledby | The `aria-labelledby` property indicates the floating label element of the MaskedTextBox. |

Ensuring accessibility

The MaskedTextBox component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the MaskedTextBox component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the MaskedTextBox component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Style appearance in EJ2 JavaScript Maskedtextbox control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of MaskedTextBox wrapper element

Use the following CSS to customize the appearance of wrapper element.

`

/ To specify height, font size, and border /

```
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input, .e-input-group textarea.e-input, .e-input-group.e-control-wrapper textarea.e-input {
```

```
font-size: 20px;
```

```
border-color: red;
```

```
height: 40px;
```

```
border: 2px solid;
```

```
}
```

`

Customizing the MaskedTextBox element on hovering

Use the following CSS to customize the Input Mask element on hovering

`

/ To specify border /

```
.e-input-group input.e-input, .e-input-group input.e-input:hover:not(.e-success):not(.e-warning):not(.e-error):not([disabled]):not(:focus), .e-input-group.e-control-wrapper input.e-input,.e-input-group.e-
```

```
control-wrapper input.e-input:hover:not(.e-success):not(.e-warning):not(.e-error):not([disabled]):no(:focus){
```

```
border: 3px solid red;
```

```
}
```

```
,
```

How To

Perform custom validation using form validator in EJ2 JavaScript Maskedtextbox control

To perform custom validation on the MaskedTextBox use the FormValidator along with custom validation rules.

In the following example, the MaskedTextBox is validated for invalid mobile number by adding custom validation in the rules collection of the FormValidator.

INDEX.TS

```
import { MaskedTextBox } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
// initializes the MaskedTextBox component
let mask: MaskedTextBox = new MaskedTextBox({
    // sets mask format to the MaskedTextBox
    mask: '000-000-0000',
    placeholder: 'Mobile Number',
    floatLabelType: 'Always'
});
mask.appendTo('#mask1');
//initialize button
let button: Button = new Button();
button.appendTo('#submit_btn');
// checks the length of mask value and returns corresponding boolean value
let customFn: (args: { [key: string]: string }) => boolean = (args: { [key: string]: string }) => {
    let argsLength:number = args.element.ej2_instances[0].value.length;
    if(argsLength != 0) {
        return argsLength >= 10;
    } else {
        return true;
    }
};
//value is returned based on the length of mask
let custom: (args: { [key: string]: string }) => boolean = (args: { [key: string]: string }) => {
    let argsLength:number = args.element.ej2_instances[0].value.length;
    if(argsLength == 0) {
        return 0;
    } else {
        return argsLength;
    }
};
// sets required property in the FormValidator rules collection
let options: FormValidatorModel = {
    rules: {
```

```

        'mask_value': { numberValue: [customFn, 'Enter valid mobile number']
    },
    },
}
// defines FormValidator to validate the MaskedTextBox
let formObject: FormValidator = new FormValidator('#form-element', options);
//FormValidator rule is added for empty MaskedTextBox
formObject.addRules('mask_value', { maxLength: [custom, 'Enter mobile
number'] });
// places error label outside the MaskedTextBox using the customPlacement
event of FormValidator
let customPlace: (element: HTMLElement, error: HTMLElement) => void =
(element: HTMLElement, error: HTMLElement) => {
    document.querySelector(".form-group").appendChild(error);
};
formObject.customPlacement = customPlace;
document.getElementById('submit_btn').onclick = () => {
    // validates the MaskedTextBox
    formObject.validate("mask_value");
    let ele: HTMLInputElement =
<HTMLInputElement>document.getElementById('mask1');
    // checks for incomplete value and alerts the format submit
    if (ele.value !== "" && ele.value.indexOf(mask.promptChar) === -1) {
        alert("Submitted");
    }
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 MaskedTextBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript MaskedTextBox Component
With Form Validation">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <form id="form-element" class="form-horizontal">
                <div class="form-group">

```

```

        <br><div><input id="mask1" type="text"
name="mask_value" class="form-control"></div>
        <button type="button" id="submit_btn" style="margin-
top: 10px">Submit</button>
    </div>
</form>
</div>
</div>
<style>
.e-mask.e-control-wrapper {
    margin-bottom: 20px;
}
label.e-error {
    margin-top: -50px;
}
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 240px;
}

```

Set cursor position while focus on the input textbox in EJ2 JavaScript Maskedtextbox control

By default, on focusing the MaskedTextBox the entire mask gets selected. You can customize by using any one of the following methods:

- Setting cursor position at the start of the MaskedTextBox.
- Setting cursor position at the end of the MaskedTextBox.
- Setting cursor at the specified position in the MaskedTextBox.

The **selectionStart** and **selectionEnd** set to **0** instead of the input element value's length, when we focus on a MaskedTextBox control filled with all mask characters. This is the default behavior of the HTML 5 input element.

Following is an example that demonstrates the above cases to set cursor position in the MaskedTextBox using [focus](#) event.

INDEX.TS

```
import { MaskedTextBox } from '@syncfusion/ej2-inputs';
// initializes the First MaskedTextBox component
let mask: MaskedTextBox = new MaskedTextBox({
    // Default MaskedTextBox
    mask: '00000-00000',
    value: '93828-3213',
    placeholder: 'Default cursor position',
    floatLabelType: 'Always'
});
mask.appendTo('#mask1');
// initializes the Second MaskedTextBox component
let mask1: MaskedTextBox = new MaskedTextBox({
    // sets mask format to the MaskedTextBox
    mask: '00000-00000',
    value: '83929-4343',
    placeholder: 'Cursor positioned at start',
    floatLabelType: 'Always',
    focus: function(args) {
        //sets cursor position at start of MaskedTextBox
        args.selectionEnd = args.selectionStart = 0;
    }
});
mask1.appendTo('#mask2');
// initializes the Third MaskedTextBox component
let mask2: MaskedTextBox = new MaskedTextBox({
    // sets mask format to the MaskedTextBox
    mask: '00000-00000',
    value: '83929-3213',
    placeholder: 'Cursor positioned at end',
    floatLabelType: 'Always',
    focus: function(args) {
        //sets cursor position at end of MaskedTextBox
        args.selectionStart = args.selectionEnd = args.maskedValue.length;
    }
});
mask2.appendTo('#mask3');
// initializes the Fourth MaskedTextBox component
let mask3: MaskedTextBox = new MaskedTextBox({
    // sets mask format to the MaskedTextBox
    mask: '+1 000-000-0000',
    value: '234-432-432',
    placeholder: 'Cursor at specified position',
    floatLabelType: 'Always',
    focus: function(args) {
        //sets cursor at specified position
        args.selectionStart = 3;
        args.selectionEnd = 3;
    }
})
```

```
});
mask3.appendTo('#mask4');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 MaskedTextBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript MaskedTextBox Component
With Different Cursor Positions">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <div class="form-group">
        <br><input id="mask1" type="text" name="mask_value1"
class="form-control">
        <input id="mask2" type="text" name="mask_value2"
class="form-control">
        <input id="mask3" type="text" name="mask_value3"
class="form-control">
        <input id="mask4" type="text" name="mask_value4"
class="form-control">
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
```

```

color: #008cff;
font-family: 'Helvetica Neue','calibiri';
font-size: 14px;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.wrap {
margin: 0 auto;
width: 240px;
}
.e-widget {
padding-bottom: 12px;
}

```

Display numeric keypad when focus on mobile devices in EJ2 JavaScript Maskedtextbox control

By default, on focusing the MaskedTextBox, alphanumeric keypad will be displayed on mobile devices. Sometimes only numeric keypad for number values is needed, and this can be achieved by setting "type" property to tel.

Refer to the following example to enable numeric keypad in MaskedTextBox.

INDEX.TS

```

import { MaskedTextBox } from '@syncfusion/ej2-inputs';
// initializes the MaskedTextBox component
let mask: MaskedTextBox = new MaskedTextBox({
    // sets mask format to the MaskedTextBox
    mask: '999-99999',
    value: "342-45432",
});
mask.appendTo('#mask1');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 MaskedTextBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript MaskedTextBox Component With Numeric Keypad For Mobile Devices">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>

```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div class="form-group">
                <br><input id="mask1" type="tel" name="mask_value"
class="form-control">
            </div>
        </div>
    </div>
</body>
</html>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 240px;
}

```

Customize the ui appearance of the control in EJ2 JavaScript Maskedtextbox control

The appearance of the MaskedTextBox can be changed by adding custom `cssClass` to the component and enabling styles.

Refer to the following example to change the appearance of the MaskedTextBox.

INDEX.TS

```

import { MaskedTextBox } from '@syncfusion/ej2-inputs';
// initializes the MaskedTextBox component
let mask: MaskedTextBox = new MaskedTextBox({
    // sets mask format to the MaskedTextBox
    mask: '00000',
    value: "42648",

```

```

placeholder: 'Enter User ID',
floatLabelType: 'Always',
cssClass: 'e-style',
focus: function(args) {
    //sets cursor position at start of MaskedTextBox
    args.selectionEnd= args.selectionStart;
}
});
mask.appendTo('#mask1');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 MaskedTextBox</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="TypeScript MaskedTextBox Component
With CustomCss Class" />
    <meta name="author" content="Syncfusion" />
    <link href="styles.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
    <script src="systemjs.config.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id='loader'>Loading....</div>
    <div id='container'>
        <div class='wrap'>
            <br/><input id="mask1" name="mask_value" class="form-control"
/>
        </div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if(ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Ej1 api migration in EJ2 JavaScript Maskedtextbox control

This article describes the API migration process of MaskEdit component from Essential JS 1 to Essential JS 2.

Common

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Adding custom class | **Property:** *cssClass*

\$("#mask").ejMaskEdit({
maskFormat: "9999",
cssClass: "custom"
}); | **Property:** *cssClass*

var mask = new ej.inputs.MaskedTextBox({
mask: "9999",
cssClass: "custom"
});
mask.appendTo("#mask"); |

| Destroy editor | Not Applicable | **Property:** *destroy*

var mask = new ej.inputs.MaskedTextBox({
mask: "00-000"
});
mask.appendTo("#mask");
mask.destroy(); |

| Disable the maskedit control | **Method:** *disable*

\$("#mask").ejMaskEdit({
maskFormat: "0000",
value: "1234"
});
var maskObj = \$("#mask").data("ejMaskEdit");
maskObj.disable(); | **Can be achieved using
API:** *enabled*
var mask = new ej.inputs.MaskedTextBox({
mask: "99,999"
});
mask.appendTo("#mask");
var maskObj = document.getElementById("mask").ej2_instances[0];
maskObj.enabled= false; |

| Enable the maskedit control | **Method:** *enable*

\$("#mask").ejMaskEdit({
maskFormat: "0000",
value: "1234"
});
var maskObj = \$("#mask").data("ejMaskEdit");
maskObj.enable(); | **Can be achieved using
API:** *enabled*
var mask = new ej.inputs.MaskedTextBox({
mask: "99,999"
});
mask.appendTo("#mask");
var maskObj = document.getElementById("mask").ej2_instances[0];
maskObj.enabled= true; |

| Control state | **Property:** *enabled*

\$("#mask").ejMaskEdit({
maskFormat: "00-000",
enabled: false
}); | **Property:** *enabled*

var mask = new ej.inputs.MaskedTextBox({
mask: "00-000",
enabled: false
});
mask.appendTo("#mask"); |

| Persistence | **Property:** *enablePersistence*

\$("#mask").ejMaskEdit({
maskFormat: "0000",
enablePersistence: true
}); | **Property:** *enablePersistence*

var mask = new ej.inputs.MaskedTextBox({
mask: "0000",
enablePersistence: true
});
mask.appendTo("#mask"); |

| Triggers when editor is focused in | **Event:** *focusIn*

\$("#mask").ejMaskEdit({
maskFormat: "00-00",
focusIn: function() {}
}); | **Event:** *focus*

var mask = new ej.inputs.MaskedTextBox({
mask: "0000",
focus: function() {}
});
mask.appendTo("#mask"); |

| Triggers when editor is focused out | **Event:** *focusOut*

\$("#mask").ejMaskEdit({
maskFormat: "0000",
focusOut: function() {}
}); | **Event** *blur*

var mask = new ej.inputs.MaskedTextBox({
mask: "00000",
blur: function() {}
});
mask.appendTo("#mask"); |

| Sets height | **Property:** *height*

\$("#mask").ejMaskEdit({
maskFormat: "0000",
height : "30px"
}); | **Can be achieved using,**

var mask = new ej.inputs.MaskedTextBox({
mask: "0000",
cssClass: "custom"
});
mask.appendTo("#mask");
Css
.e-maskedtextbox.custom{
height: 30px;
} |

| HTML Attributes | **Property:** *htmlAttributes*
 "0000",
htmlAttributes: {name: "maskedtextbox"}
}); | **Can be achieved using**
HTML
 <input id="mask" type="text" name="maskedtextbox" />
var mask = new
 ej.inputs.MaskedTextBox({
mask: "0000"
});
mask.appendTo("#mask"); |

| Specifies Input mode | **Property:** *inputMode*
 "0000",
inputMode: ej.InputMode.Password
}); | **Can be achieved using**
HTML
 <input id="mask" type="password" />
var mask = new
 ej.inputs.MaskedTextBox({
mask: "0000",
});
mask.appendTo("#mask"); |

| Triggers on key press | **Event:** *keyPress*
 "000",
keyPress: function() {}
}); | **Can be achieved using native event** **HTML**
 <input id="mask" type="text" onkeypress="keyPress()" />
 <script>
function
 keyPress(){}
</script>
 |

| Triggers on key up | **Event:** *keyUp*
 "9999",
keyUp: function() {}
}); | **Can be achieved using native event** **HTML**
 <input id="mask" type="text" onkeyup="keyUp()" />
 <script>
function keyUp(){}
</script>
 |

| Triggers on mouse out in maskedit control | **Event:** *mouseOut*
 "0000-000",
mouseOut: function() {}
}); | **Can be achieved using native event** **HTML**
 <input id="mask" type="text" onmouseout="mouseOut()" />
 <script>
function mouseOut(){}
</script>
 |

| Triggers when mouse over in maskedit control | **Event:** *mouseOver*
 "00-00",
mouseOver: function() {}
}); | **Can be achieved using native event** **HTML**
 <input id="mask" type="text" onmouseover="mouseOver()" />
 <script>
function mouseOver(){}
</script>
 |

| Name of maskedit control | **Property:** *name*
 "0000",
name: "pin"
}); | **Can be achieved using** **HTML**
 <input id="mask" type="text" name="maskedtextbox" />
var mask = new ej.inputs.MaskedTextBox({
mask:
 "0000",
});
mask.appendTo("#mask"); |

| Triggers on keydown | **Event:** *onKeyDown*
 "9999-9999",
onKeyDown: function() {}
}); | **Can be achieved using native event**
HTML
 <input id="mask" type="text" onkeydown="keyDown()" />
 <script>
function
 keyDown(){}
</script>
 |

| Read only | **Property:** *readOnly*
 "99-999",
readOnly: true
}); | **Can be achieved using**,
var mask = new
 ej.inputs.MaskedTextBox({
mask: "0000",
enabled: false,

});
mask.appendTo("#mask");
CSS
 .e-mask{
 background-image: none
 !important;
border-bottom-color: rgba(0, 0, 0, 0.42) !important;
 |

| Right to left | **Property:** *textAlign*
 "9999",
textAlign: "right"
}); | **Property:** *enableRtl*

var mask = new
 ej.inputs.MaskedTextBox({
mask: "9999",
enableRtl:
 true
});
mask.appendTo("#mask"); |

```
| Sets width | Property: width<br/><br/>$("#mask").ejMaskEdit({<br/>maskFormat: "0000",<br/>width:
"100px"<br/>}); | Property: width<br/><br/>var mask = new ej.inputs.MaskedTextBox({<br/>mask:
"0000",<br/>width: "100px"<br/>});<br/>mask.appendTo("#mask"); |
```

Mask Configuration

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

```
| Triggers on value change | Event change<br/><br/>$("#mask").ejMaskEdit({<br/>maskFormat: "00-00",<br/>change: function() {}<br/>}); | Event: change<br/><br/>var mask = new ej.inputs.MaskedTextBox({<br/>mask: "00-00",<br/>change: function() {}<br/>});<br/>mask.appendTo("#mask"); |
```

```
| Clears masked text/value | Method: clear<br><br>$("#mask").ejMaskEdit({<br>maskFormat: "0000",<br>value: "1234"<br>});<br>var maskObj = $("#mask").data("ejMaskEdit");<br>maskObj.clear(); | Can be achieved using<br>var mask = new ej.inputs.MaskedTextBox({<br>mask: "99,999"<br>});<br>mask.appendTo("#mask");<br>var maskObj = document.getElementById("mask").ej2_instances[0];<br>maskObj.value = ""; |
```

```
| Triggers on creation | Event: create<br/><br/>$("#mask").ejMaskEdit({<br/>maskFormat: "00-00",<br/>create: function() {}<br/>}); | Event: created<br/><br/>var mask = new ej.inputs.MaskedTextBox({<br/>mask: "00-00",<br/>created: function() {}<br/>});<br/>mask.appendTo("#mask"); |
```

```
| Custom Character | Property: customCharacter<br/><br/>$("#mask").ejMaskEdit({<br/>maskFormat:
"C-0000",<br/>customCharacter : "#"<br/>}); | Property: customCharacters<br/><br/>var mask = new
ej.inputs.MaskedTextBox({<br/>mask: "C-0000",<br/>customCharacters: {C:
'#'}<br/>});<br/>mask.appendTo("#mask"); |
```

```
| Triggers when maskedit control is destroyed | Event:  
destroy<br/><br/>$("#mask").ejMaskEdit({<br/>maskFormat: "00-00",<br/>destroy: function() {}<br/>});  
| Event: destroyed<br/><br/>var mask = new ej.inputs.MaskedTextBox({<br/>mask: "00-  
00",<br/>destroyed: function() {}<br/>});<br/>mask.appendTo("#mask"); |
```

```
| Placeholder float type | Not Applicable | Property: floatLabelType<br/><br/>var mask = new
ej.inputs.MaskedTextBox({<br/>mask: "9,999",<br/>placeholder: "Enter value",<br/>floatLabelType:
"Auto"<br/>});<br/>mask.appendTo("#mask"); |
```

| Gets pure value of masked edit control | **Method:**

```
getStrippedValue<br/><br/>$("#mask").ejMaskEdit({<br/>maskFormat: "0000",<br/>value:
"123"<br/>});<br/>var maskObj = $("#mask").data("ejMaskEdit");<br/>maskObj.getStrippedValue(); |
```

Can be achieved using,
 var mask = new ej.inputs.MaskedTextBox({
mask: "00-00",
value: "1234"
});
mask.appendTo("#mask");
var maskObj = document.getElementById("mask").ej2_instances[0];
alert(maskObj.value);
 |

```
| Get whole maskedit value | Method:  
getUnstrippedValue<br/><br/>$(("#mask").ejMaskEdit({<br/>maskFormat: "00-00",<br/>value:  
"1234"<br/>}));<br/>var maskObj = $(("#mask").data("ejMaskEdit"));<br/>maskObj.getUnstrippedValue();  
| Method: getMaskedValue<br/><br/>var mask = new ej.inputs.MaskedTextBox({<br/>mask: "00-  
00",<br/>value: "1234"<br/>});<br/>mask.appendTo("#mask");<br/>mask.getMaskedValue(); |
```

| Hides prompt character on focus out | **Property:**
hidePromptOnLeave

\$("#mask").eiMaskEdit({
maskFormat:

"aaaa",
hidePromptOnLeave: true
}); | **Can be achieved using**
var mask = new ej.inputs.MaskedTextBox({
mask: '000-000-0000',
focus: function(){
 this.promptChar = "";
 }
});
var maskObj = document.getElementById("mask1")
maskObj.addEventListener("focusout",function(){
maskObj.ej2instances[0].promptChar = " ";
 } |

| Mask format | **Property:** maskFormat

\$("#mask").ejMaskEdit({
maskFormat: "99,999"
}); | **Property:** mask

var mask = new ej.inputs.MaskedTextBox({
mask: "99,999"
});
mask.appendTo("#mask"); |

| Prompt character | Not Applicable | **Property:** promptChar

var mask = new ej.inputs.MaskedTextBox({
mask: "0000",
promptChar: "#"
});
mask.appendTo("#mask"); |

| Clear Button | Not Applicable | **Property:** promptChar

var mask = new ej.inputs.MaskedTextBox({
mask: "aaaa",
showClearButton: true
});
mask.appendTo("#mask"); |

| Prompt character display | **Property:** showPromptChar

\$("#mask").ejMaskEdit({
maskFormat: "\$ 99-999",
showPromptChar: false
}); | **Can be achieved using**
var mask = new ej.inputs.MaskedTextBox({
mask: "0000",
promptChar: "
");
mask.appendTo("#mask"); |

| Show rounded corner | **Property:** showRoundedCorner

\$("#mask").ejMaskEdit({
maskFormat: "0000",
showRoundedCorner: true
}); | **Can be achieved using following Css**
var mask = new ej.inputs.MaskedTextBox({
mask: "9999",
cssClass: "e-style",
 floatLabelType: "Always"
});
mask.appendTo("#mask");
CSS
#mask{
border: 2px solid grey;
padding: 10px;
border-radius: 10px;
}
.e-control-wrapper.e-mask.e-float-input.e-style .e-float-line::before, .e-control-wrapper.e-mask.e-float-input.e-style .e-float-line::after {
 background: none ;
}
</style> |

| Value of maskedit control | **Property:** value

\$("#mask").ejMaskEdit({
maskFormat: "0000",
value: "1234"
}); | **Property:** value

var mask = new ej.inputs.MaskedTextBox({
mask: "0000",
value: "1234"
});
mask.appendTo("#mask"); |

| Displays hint on maskedit control | **Property:** watermarkText

\$("#mask").ejMaskEdit({
maskFormat: "9999",
watermarkText: "Enter value"
}); | **Property:** placeholder

var mask = new ej.inputs.MaskedTextBox({
mask: "9999",
placeholder: "Enter value"
});
mask.appendTo("#mask"); |

Validation

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Displays error until correct value is entered | **Property:** showError

\$("#mask").ejMaskEdit({
maskFormat: "99-999",
showError: true
}); | **MaskedTextBox by default shows error until correct value is entered.**
var mask = new

```
ej.inputs.MaskedTextBox({<br/>mask: "9999",<br/>placeholder: "Enter
value"<br/>});<br/>mask.appendTo("#mask"); |
```

| Validation message | **Property:**

```
validationMessage<br/><br/>$("#mask").ejMaskEdit({<br/>maskFormat: "0000",<br/>validationRules:
{required: true},<br/>validationMessage: {required: "Required value"}<br/>}); | Validation in
```

MaskedTextBox can be achieved through form validation
var options = {
 rules:

```
{<br/>'maskValue': { required: [ true, 'Enter valid mobile number' ] },<br/> },<br/>}<br/>var formObject =
new ej.inputs.FormValidator('#form-element', options);<br/> formObject.customPlacement =
function(element, error) {<br/> document.querySelector(".form-
```

```
group").appendChild(error);<br/>}<br/>var mask = new ej.inputs.MaskedTextBox({<br/> mask: '000-
000-0000',<br/>});<br/>mask.appendTo("#mask");<br/>HTML <br/> <form id="form-element"
class="form-horizontal"><br/><div class = "form-group"><br/><input id="mask" type="text"
name="maskValue" class="form-control" /> <br/> <div id="error"></div><br/></div><br/></form> |
```

| Validation Rules | **Property:** validationRules

\$("#mask").ejMaskEdit({
maskFormat: "00-

```
00",<br/>validationRules: {required: true}<br/>}); | Validation in MaskedTextBox can be achieved
through form validation<br/>var options = {<br/> rules: {<br/>'maskValue': { required: [ true] },<br/>
},<br/>}<br/>var formObject = new ej.inputs.FormValidator('#form-element', options);<br/>
```

```
formObject.customPlacement = function(element, error) {<br/> document.querySelector(".form-
group").appendChild(error);<br/>}<br/>var mask = new ej.inputs.MaskedTextBox({<br/> mask: '000-
000-0000',<br/>});<br/>mask.appendTo("#mask");<br/>HTML <br/> <form id="form-element"
class="form-horizontal"><br/><div class = "form-group"><br/><input id="mask" type="text"
name="maskValue" class="form-control" /> <br/> <div id="error"></div><br/></div><br/></form> |
```

Mention

Working with data in EJ2 JavaScript Mention control

The Mention loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of either [array](#) or [DataManager](#).

The Mention also supports different kinds of data services such as OData V4 and Web API, and data formats such as XML, JSON, and JSONP with the help of [DataManager](#) adaptors.

| Fields | Type | Description |

| ----- | ----- | ----- |

| text | [string](#) | Specifies the display text of each list item. |

| value | [number or string](#) | Specifies the hidden data value mapped to each list item that should contain a unique value. |

| groupBy | [string](#) | Specifies the category under which the list item has to be grouped. |

| iconCss | [string](#) | Specifies the icon class of each list item. |

When binding complex data to the Mention, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in three ways as described in the following.

Array of simple data

The Mention has provided support to load an array of primitive data such as strings and numbers. Here, both the value and text fields act the same.

INDEX.TS

```
import { Mention } from '@syncfusion/ej2-dropdowns';
// defined the array of data
let userData: string[] = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
// initialize Mention control
let mentionObject: Mention = new Mention({
    //set the data to dataSource property
    dataSource: userData
});
// render initialized Mention
mentionObject.appendTo('#mentionElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Mention</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/bootstrap5.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/bootstrap5.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="width:200px;">
    <!--element which is the Mention target to list the suggestions-->
    <label style="font-size: 15px; font-weight: 600;">Comments</label>
    <div id="mentionElement" style="min-height: 100px; border: 1px solid
    #D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
    600px;"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
```



```
</body></html>
```

Array of JSON data

The Mention can generate its list of items through an array of JSON data. Therefore the appropriate columns should be mapped to the [fields](#) property.

In the following example, ID column and Game column from complex data have been mapped to the value field and text field, respectively.

INDEX.TS

```
import { Mention } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
  { ID: 'game1', Game: 'Badminton' },
  { ID: 'game2', Game: 'Football' },
  { ID: 'game3', Game: 'Tennis' },
  { ID: 'game4', Game: 'Hockey' },
  { ID: 'game5', Game: 'Basketball' }
];
//initiate the Mention
let mentionObject: Mention = new Mention({
  // bind the sports Data to datasource property
  dataSource: sportsData,
  // maps the appropriate column to fields property
  fields: { text: 'Game', value: 'ID' }
});
//render the control
mentionObject.appendTo('#mentionElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Mention</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="width:200px;">
```

```

        <label style="font-size: 15px; font-weight: 600;">Comments</label>
        <!--element which is the Mention target to list the suggestions-->
        <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Array of complex data

The Mention can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Code.ID** column and **Country.Name** column from complex data have been mapped to the **value** field and **text** field, respectively.

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let CountryGroup: { [key: string]: Object }[] = [
    { Country: { Name: 'Australia' }, Code: { ID: 'AU' } },
    { Country: { Name: 'Bermuda' }, Code: { ID: 'BM' } },
    { Country: { Name: 'Canada' }, Code: { ID: 'CA' } },
    { Country: { Name: 'Cameroon' }, Code: { ID: 'CM' } },
    { Country: { Name: 'Denmark' }, Code: { ID: 'DK' } },
    { Country: { Name: 'France' }, Code: { ID: 'FR' } }
];
//initiate the Mention
let mentionObject: Mention = new Mention({
    // bind the sports Data to datasource property
    dataSource: CountryGroup,
    // maps the appropriate column to fields property
    fields: { text: 'Country.Name', value: 'Code.ID' }
});
//render the control
mentionObject.appendTo('#mentionElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Mention</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="width:200px;">
        <label style="font-size: 15px; font-weight: 600;">Comments</label>
        <!--element which is the Mention target to list the suggestions-->
        <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Binding remote data

The Mention supports retrieval of data from remote data services with the help of **DataManager** control. The [Query](#) property is used to fetch the data from the database and bind it to the Mention control.

OData v4 adaptor - Binding OData v4 service

The ODataV4 is an improved version of OData protocols, and the **DataManager** can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the [odata documentation](#). To bind OData v4 service, use the **ODataV4Adaptor**.

The following sample displays the first 6 contacts from **Customers** table of the **Northwind** Data Service.

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the control
let mentionObject: Mention = new Mention({
    //bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property

```

```

    query: new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'ContactName', value: 'CustomerID' },
    popupWidth: '250px'
});
//render the control
mentionObject.appendTo('#mentionElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Mention</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="width:200px;">
    <label style="font-size: 15px; font-weight: 600;">Comments</label>
    <!--element which is the Mention target to list the suggestions-->
    <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Web API adaptor

You can use **WebApiAdaptor** to bind mention with Web API created using OData endpoint.

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes

```

```
import { Query, DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
//initiates the control
let mentionObject: Mention = new Mention({
    //bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://ej2services.syncfusion.com/production/web-
services/api/Employees',
        adaptor: new WebApiAdaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().select(['FirstName', 'EmployeeID']).take(7),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },
    popupWidth: '250px'
});
//render the control
mentionObject.appendTo('#mentionElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Mention</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="width:200px;">
    <label style="font-size: 15px; font-weight: 600;">Comments</label>
    <!--element which is the Mention target to list the suggestions-->
    <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See also

- [Customization](#)
- [How to perform filtering](#)

Mention integration in EJ2 JavaScript Rich text editor control

By integrating the [Mention](#) control with a Rich Text Editor, users can easily mention or tag other users or objects from the suggested list without having to manually type out their names or other identifying information.

The [target](#) property of the Mention control allows you to specify the **ID** of the content editable div element within the Rich Text Editor that you want to bind the Mention control to. This allows you to enable the Mention functionality within the Rich Text Editor, so that users can mention or tag other users or objects from the suggested list while editing the text.

When the user types the **@** symbol followed by a character, the Rich Text Editor will display a list of suggestions for items that the user can select from. The user can then select an item from the list by clicking on it, or by typing the name of the item they want to tag.

In the following sample, configured the following properties with popup dimensions.

- [allowSpaces](#) - Allow to continue search action if user enter space after mention character while searching.
- [suggestionCount](#) - The maximum number of items that will be displayed in the suggestion list.
- [itemTemplate](#) - Used to display the customized appearance in suggestion list.

INDEX.TS

```
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
import { Mention } from '@syncfusion/ej2-dropdowns';
let emailData: { [key: string]: Object }[] = [
    { Name: "Selma Rose", Status: "active", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/2.png",
      EmailId: "selma@gmail.com" },
    { Name: "Maria", Status: "active", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/1.png",
      EmailId: "maria@gmail.com" },
    { Name: "Russo Kay", Status: "busy", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/8.png",
      EmailId: "russo@gmail.com" },
    { Name: "Camden Kate", Status: "active", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/9.png",
      EmailId: "camden@gmail.com" },
    { Name: "Robert", Status: "busy", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/dp.png",
      EmailId: "robert@gmail.com" },
  ],
```

```

    { Name: "Garth", Status: "active", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/7.png",
      EmailId: "garth@gmail.com" },
    { Name: "Andrew James", Status: "away", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/pic04.png",
      EmailId: "andrew@gmail.com" },
    { Name: "Olivia", Status: "busy", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/5.png",
      EmailId: "olivia@gmail.com" },
    { Name: "Sophia", Status: "away", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/6.png",
      EmailId: "sophia@gmail.com" },
    { Name: "Margaret", Status: "active", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/3.png",
      EmailId: "margaret@gmail.com" },
    { Name: "Ursula Ann", Status: "active", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/dp.png",
      EmailId: "ursula@gmail.com" },
    { Name: "Laura Grace", Status: "away", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/4.png",
      EmailId: "laura@gmail.com" },
    { Name: "Albert", Status: "active", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/pic03.png",
      EmailId: "albert@gmail.com" },
    { Name: "William", Status: "away", EmployeeImage:
      "https://ej2.syncfusion.com/demos/src/rich-text-editor/images/8.png",
      EmailId: "william@gmail.com" }
  ];
  let defaultRTE: RichTextEditor = new RichTextEditor({
    placeholder: 'Type @ and tag the name',
    actionBegin: (args) => {
      if (args.requestType === 'EnterAction') {
        args.cancel = true;
      }
    }
  });
  defaultRTE.appendTo('#mention_integration');
  // Initialize Mention control.
  let emailObj: Mention = new Mention({
    dataSource: emailData,
    fields: { text: 'Name' },
    suggestionCount: 8,
    displayTemplate: '<a href=mailto:${EmailId}
title=${EmailId}>@${Name}</a>',
    itemTemplate: '<table><tr><td><div id="mention-TemplateList"><span
class="e-badge e-badge-success e-badge-overlap e-badge-dot e-badge-bottom
${Status}></span></div></td><td><span class="person">${Name}</span><span
class="email">${EmailId}</span></td></tr></table>',
    popupWidth: '250px',
    popupHeight: '200px',
    target: '#mention_integration_rte-edit-view',
    allowSpaces: true
  });
  emailObj.appendTo('#mentionEditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="mentionEditor"></div>
    <div id="mention_integration">
      <p>Hello <span contenteditable="false" class="e-mention-
chip"><a href="mailto:maria@gmail.com"
title="maria@gmail.com">@Maria</a></span>,</p>
      <p>Welcome to the mention integration with rich text editor
demo. Type <code>@</code> character and tag user from the suggestion list.
</p>
    </div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


[View Sample](#)[See Also](#)

- [Mention](#)

Filtering data in EJ2 JavaScript Mention control

The Mention control has built-in support to filter data items. The filter operation starts as soon as you start typing characters in the mention element.

Limit the minimum filter character

You can control the minimum length of user input to initiate the search action using the [minLength](#) property. This can be useful if you have a very large list of data. The default value is 0, where the suggestion list opens as soon as the user inputs the mention character.

The remote request does not fetch the search data until the search key contains three characters as shown in the following example.

INDEX.TS

```
import { Mention } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let searchData: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let filter: Mention = new Mention({
    dataSource: searchData,
    query: new Query().select(['ContactName', 'CustomerID']).take(7),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    minLength: 3
});
filter.appendTo('#mentionElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Mention</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/bootstrap5.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/bootstrap5.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="width:200px;">
        <!--element which is the Mention target to list the suggestions-->
        <label style="font-size: 15px; font-weight: 600;">Comments</label>
        <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change the filter type

While filtering, you can change the filter type to **Contains**, **StartsWith**, or **EndsWith** in the [filterType](#) property. The default filter operator is **Contains**.

- **StartsWith** - Filter the items that begin with the specified text value.
- **Contains** - Filter the items that contain the specified text value.
- **EndsWith** - Filter the items that end with the specified text value.

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let searchData: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let filter: Mention = new Mention({
    dataSource: searchData,
    query: new Query().select(['ContactName', 'CustomerID']).take(7),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    filterType: 'StartsWith'
});
filter.appendTo('#mentionElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Mention</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/bootstrap5.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/bootstrap5.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="width:200px;">
        <!--element which is the Mention target to list the suggestions-->
        <label style="font-size: 15px; font-weight: 600;">Comments</label>
        <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Allow spacing between search

While filtering the data in the data source, you can allow the space in the middle of the mention using the [allowSpaces](#) property. If the data source does not match with the mentioned element data, the popup will be hidden on the space key press. The default value of the [allowSpaces](#) is `false`.

By default, the [allowSpaces](#) property is disabled, and the space ends the mention control search.

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
// defined the array of complex data
let employeeData: { [key: string]: Object }[] = [
    { Name: 'Andrew Fuller', ID: '1' },
    { Name: 'Anne Dodsworth', ID: '2' },
    { Name: 'Janet Leverling', ID: '3' },
    { Name: 'Laura Callahan', ID: '4' },
    { Name: 'Margaret Peacock', ID: '5' }
];
let filter: Mention = new Mention({
    dataSource: employeeData,
    // maps the appropriate column to fields property
    fields: { text: 'Name', value: 'ID' },
    allowSpaces: true,

```

```
});
filter.appendTo('#mentionElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Mention</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/bootstrap5.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/bootstrap5.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="width:200px;">
    <!--element which is the Mention target to list the suggestions-->
    <label style="font-size: 15px; font-weight: 600;">Comments</label>
    <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customize the suggestion item count

While filtering, you can customize the number of list items to be displayed in the suggestion list using the [suggestionCount](#) property.

INDEX.TS

```
import { Mention } from '@syncfusion/ej2-dropdowns';
let emailData: { [key: string]: Object }[] = [
  { Name: "Selma Rose", EmailId: "selma@gmail.com" },
  { Name: "Maria", EmailId: "maria@gmail.com" },
  { Name: "Russo Kay", EmailId: "russo@gmail.com" },
  { Name: "Robert", EmailId: "robert@gmail.com" },
```

```

    { Name: "Camden Kate", EmailId: "camden@gmail.com" },
    { Name: "Garth", EmailId: "garth@gmail.com" },
    { Name: "Andrew James", EmailId: "james@gmail.com" },
    { Name: "Olivia", EmailId: "olivia@gmail.com" },
    { Name: "Sophia", EmailId: "sophia@gmail.com" },
    { Name: "Margaret", EmailId: "margaret@gmail.com" },
    { Name: "Ursula Ann", EmailId: "ursula@gmail.com" },
    { Name: "Laura Grace", EmailId: "laura@gmail.com" },
    { Name: "Albert", EmailId: "albert@gmail.com" },
    { Name: "William", EmailId: "william@gmail.com" }
  ];
  let filter: Mention = new Mention({
    dataSource: emailData,
    fields: { text: 'Name' },
    suggestionCount: 8
  });
  filter.appendTo('#mentionElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Mention</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/bootstrap5.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/bootstrap5.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="width:200px;">
    <!--element which is the Mention target to list the suggestions-->
    <label style="font-size: 15px; font-weight: 600;">Comments</label>
    <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

See also

- [Templates](#)

Sorting in EJ2 JavaScript Mention control

You can display the suggestion list items in a specific order. It has possible types as **Ascending**, **Descending** and **None** in the [sortOrder](#) property.

- **None** - The data source is not sorted.
- **Ascending** - The data source is sorted in ascending order.
- **Descending** - The data source is sorted in descending order.

INDEX.TS

```
import { Mention } from '@syncfusion/ej2-dropdowns';
// defined the array of complex data
let sportsData: { [key: string]: Object }[] = [
  { ID: 'game1', Game: 'Badminton' },
  { ID: 'game2', Game: 'Football' },
  { ID: 'game3', Game: 'Tennis' },
  { ID: 'game4', Game: 'Hockey' },
  { ID: 'game5', Game: 'Basketball' }
];
//initiates the control
let mentionObject: Mention = new Mention({
  //set the data to dataSource property
  dataSource: sportsData,
  // maps the appropriate column to fields property
  fields: { text: 'Game', value: 'ID' },
  //sort the resulted items
  sortOrder: 'Descending'
});
//render the control
mentionObject.appendTo('#mentionElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Mention</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/bootstrap5.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/bootstrap5.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="width:200px;">
        <!--element which is the Mention target to list the suggestions-->
        <label style="font-size: 15px; font-weight: 600;">Comments</label>
        <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Templates in EJ2 JavaScript Mention control

The Mention has been provided with several options to customize each suggestion list item, display item, and data loading indication.

Item template

The content of each list item in the Mention can be customized using the [itemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data using [itemTemplate](#).

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the control
let mentionObject: Mention = new Mention({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },
    //set width to popup list
    popupWidth: '200px',
    //set the value to itemTemplate property

```

```

    itemTemplate:"<span><span class='name'>${FirstName}</span><span class
    ='city'>${City}</span></span>"
  });
  //render the control
  mentionObject.appendTo('#mentionElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="width:200px;">
    <label style="font-size: 15px; font-weight: 600;">Comments</label>
    <!--element which is the Mention target to list the suggestions-->
    <div id="mentionElement" style="min-height: 100px; border: 1px solid
    #D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
    600px;"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Display template

You can customize the mentioned value's display appearance using the [displayTemplate](#) property.

In the following sample, the selected value is displayed as a combined text of both **FirstName** and **City** in the mention element, which is separated by a hyphen.

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
//import data manager related classes

```



```

import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the control
let mentionObject: Mention = new Mention({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName',
    'City', 'EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },
    //set width to popup list
    popupWidth: '200px',
    //set the template value to itemTemplate property
    itemTemplate: "<span><span>${FirstName}</span><span class
    ='city'>${City}</span></span>",
    //set the value to displayTemplate property
    displayTemplate: "<span>${FirstName} - ${City}</span>"
});
//render the control
mentionObject.appendTo('#mentionElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container" style="width:200px;">
        <label style="font-size: 15px; font-weight: 600;">Comments</label>
        <!--element which is the Mention target to list the suggestions-->
        <div id="mentionElement" style="min-height: 100px; border: 1px solid
        #D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
        600px;"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

No records template

You can show the custom design of the popup list content when no data and matches are found on the search with the help of [noRecordsTemplate](#) property.

INDEX.TS

```
import { Mention } from '@syncfusion/ej2-dropdowns';
//initiates the control
let mentionObject: Mention = new Mention({
    //bind the data manager instance to dataSource property
    dataSource: [],
    //set the value to noRecords template
    noRecordsTemplate: "<span class='norecord'> NO DATA AVAILABLE</span>"
});
//render the control
mentionObject.appendTo('#mentionElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 DropDownList</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container" style="width:200px;">
        <label style="font-size: 15px; font-weight: 600;">Comments</label>
        <!--element which is the Mention target to list the suggestions-->
```

```

    <div id="mentionElement" style="min-height: 100px; border: 1px solid
    #D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
    600px;"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Spinner template

Display the customized waiting spinner, when data fetching takes time to load in the suggestion list by using the [spinnerTemplate](#) property.

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the control
let mentionObject: Mention = new Mention({
  //bind the data manager instance to dataSource property
  dataSource: new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new Query().from('Employees').select(['FirstName',
  'City', 'EmployeeID']).take(26),
  //map the appropriate columns to fields property
  fields: { text: 'FirstName', value: 'EmployeeID' },
  //set width to popup list
  popupWidth: '200px',
  //set the value to spinner template
  spinnerTemplate: '<div class="spinner_loader"></div>'
});
//render the control
mentionObject.appendTo('#mentionElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 DropDownList</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="width:200px;">
        <label style="font-size: 15px; font-weight: 600;">Comments</label>
        <!--element which is the Mention target to list the suggestions-->
        <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [How to achieve filtering](#)

Localization in EJ2 JavaScript Mention control

The Localization library allows you to localize static text content of the [noRecordsTemplate](#) properties according to the culture currently assigned to the Mention.

| Locale key | en-US (default) |

|-----|-----|

| noRecordsTemplate | No records found |

Loading translations

To load the translation object to your application, use the load function of the **L10n** class.

In the following sample, French culture is set to the mention control and no data is loaded. Hence, the [noRecordsTemplate](#) property displays its text in French culture initially.

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
// import L10n class for load function
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
// bind remotedata to showcase actionFailureTemplate in offline.

```

```

let customerData: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let mentionObject: Mention = new Mention({
    dataSource: customerData,
    // set locale culture to Mention
    locale: 'fr-BE',
    // map appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // take 0 item to showcase noRecordsTemplate property.
    query: new Query().select(['ContactName', 'CustomerID']).take(0)
});
mentionObject.appendTo('#mentionElement');
L10n.load({
    'fr-BE': {
        'mention': {
            'noRecordsTemplate': "Aucun enregistrement trouvé"
        }
    }
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Mention</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/bootstrap5.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/bootstrap5.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="width:200px;">
        <!--element which is the Mention target to list the suggestions-->
        <label style="font-size: 15px; font-weight: 600;">Comments</label>
        <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
    </div>
    <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See also

- [Accessibility](#)
- [How to bind the data to the mention](#)

Customization in EJ2 JavaScript Mention control

Show or hide mention character

You can show the mention character as the prefix of the selected item in mention component using [showMentionChar](#) property. The default value of `showMentionChar` is `false`.

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
// defined the array of complex data
let emailData: { [key: string]: Object }[] = [
    { Name: 'Selma Rose', EmailId : 'selma@gmail.com' },
    { Name: 'Maria', EmailId : 'maria@gmail.com' },
    { Name: 'Russo kay', EmailId : 'russo@gmail.com' },
    { Name: 'Robert', EmailId : 'robert@gmail.com' },
    { Name: 'Garth', EmailId : 'garth@gmail.com' }
];
// initialize Mention control
let mentionObject: Mention = new Mention({
    //set the data to dataSource property
    dataSource: emailData,
    // maps the appropriate column to fields property
    fields: { text: 'Name', value: 'EmailId' },
    mentionChar: '#',
    showMentionChar: true
});
// render initialized Mention
mentionObject.appendTo('#mentionElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Mention</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/bootstrap5.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/bootstrap5.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="width:200px;">
        <!--element which is the Mention target to list the suggestions-->
        <label style="font-size: 15px; font-weight: 600;">Comments</label>
        <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding the suffix character after selection

You can add the suffix character while selecting an item in the Mention component using [suffixText](#) property. You can add space or new line as suffix to the selected item. The default values are empty string.

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
// define the array of complex data
let sportsData: { [key: string]: Object }[] = [
    { ID: 'game1', Game: 'Badminton' },
    { ID: 'game2', Game: 'Football' },
    { ID: 'game3', Game: 'Tennis' },
    { ID: 'game4', Game: 'Hockey' },
    { ID: 'game5', Game: 'Basketball' }
];
// initialize Mention control
let mentionObject: Mention = new Mention({
    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { text: 'Game', value: 'ID' },
    suffixText: '&#160;'
});
// render initialized Mention
mentionObject.appendTo('#mentionElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Mention</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="width:200px;">
    <label style="font-size: 15px; font-weight: 600;">Comments</label>
    <!--element which is the Mention target to list the suggestions-->
    <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Configure the popup list

You can customize the suggestion list as width and height using the [popupHeight](#) and [popupWidth](#) properties.

By default, the popup list width value is set as `auto`. Depending on the mentioned suggestion data list, the width value is automatically adjusted. The popup list height value is set as `300px`.

INDEX.TS

```

import { Mention } from '@syncfusion/ej2-dropdowns';
// define the array of complex data
let countriesData: { [key: string]: Object }[] = [
  { Country : 'Australia', Code : 'AU' },
  { Country : 'Bermuda', Code : 'BM' },
  { Country : 'Canada', Code : 'CA' },
  { Country : 'Cameroon', Code : 'CM' },
  { Country : 'Denmark', Code : 'DK' }
];

```



```
// initialize Mention control
let mentionObject: Mention = new Mention({
  //set the data to dataSource property
  dataSource: countriesData,
  // maps the appropriate column to fields property
  fields: { text: 'Country', value: 'Code' },
  //set height to popup list
  popupHeight: '200px',
  //set width to popup list
  popupWidth: '250px'
});
// render initialized Mention
mentionObject.appendTo('#mentionElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Mention</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/bootstrap5.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/bootstrap5.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="width:200px;">
    <!--element which is the Mention target to list the suggestions-->
    <label style="font-size: 15px; font-weight: 600;">Comments</label>
    <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Trigger character

You can customize the trigger character by using the [mentionChar](#) property in the Mention control. The trigger character triggers the suggestion list to display in the target area.

By default, the [mentionChar](#) is @.

Accessibility in EJ2 JavaScript Mention control

Web accessibility makes web content and web applications more accessible for people with disabilities. Mention control provides built-in compliance with WAI-ARIA specifications. The WAI-ARIA support is achieved using the attributes such as [aria-selected](#) and [aria-activedescendent](#).

The Mention component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Mention component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Mention control uses the **Listbox** role where each list item has an **option** role. The following **ARIA attributes** denote the Mention state.

| Properties | Functionalities |

| --- | --- |

| aria-selected | Indicates the selected option. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

Keyboard interaction

You can use the following key shortcuts to access the Mention without interruptions.

| Keyboard shortcuts | Actions |

| --- | --- |

| Arrow Down | Selects the first item in the Mention list. Otherwise, selects the item next to the currently selected item. |

| Arrow Up | Selects the item previous to the currently selected one. |

| Esc(Escape) | Closes the popup list when it is in an open state. |

| Enter | Selects the focused item, and when it is in an open state the popup list closes. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, inserts the selected popup list item and closes the popup list. |

INDEX.TS

```
import { Mention } from '@syncfusion/ej2-dropdowns';
// defined the array of complex data
let employeeData: { [key: string]: Object }[] = [
  { Name: 'Andrew Fuller', ID: '1' },
  { Name: 'Janet Leverling', ID: '2' },
  { Name: 'Laura Callahan', ID: '3' },
  { Name: 'Margaret Peacock', ID: '4' },
  { Name: 'Anne Dodsworth', ID: '5' }
];
// initialize Mention control
let mentionObject: Mention = new Mention({
  //set the data to dataSource property
  dataSource: employeeData,
  // maps the appropriate column to fields property
```

```

    fields: { text: 'Name', value: 'ID' }
  });
  // render initialized Mention
  mentionObject.appendTo('#mentionElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Mention</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="width:200px;">
    <label style="font-size: 15px; font-weight: 600;">Comments</label>
    <!--element which is the Mention target to list the suggestions-->
    <div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Ensuring accessibility

The Mention component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Mention component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Mention component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Menu Bar

Icons and sub menu items in EJ2 JavaScript Menu control

Icons

The menu item contains an icon/image in it to provide a visual representation of an action. To place the icon on a menu item, set the [iconCss](#) property with the required icon CSS. By default, the icon is positioned at the left of the menu item. In the following sample, the icons of File and Edit menu items and Open, Save, Cut, Copy, and Paste sub menu items are added using the [iconCss](#) property.

INDEX.TS

```
import { Menu, MenuItemModel } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition
let menuItems: MenuItemModel[] = [
    {
        text: 'File',
        iconCss: 'em-icons e-file',
        items: [
            { text: 'Open', iconCss: 'em-icons e-open' },
            { text: 'Save', iconCss: 'e-icons e-save' },
            { separator: true },
            { text: 'Exit' }
        ]
    },
    {
        text: 'Edit',
        iconCss: 'em-icons e-edit',
        items: [
            { text: 'Cut', iconCss: 'em-icons e-cut' },
            { text: 'Copy', iconCss: 'em-icons e-copy' },
            { text: 'Paste', iconCss: 'em-icons e-paste' }
        ]
    },
    {
        text: 'View',
        items: [
            { text: 'Toolbar' },
            { text: 'Sidebar' },
            { text: 'Full Screen' }
        ]
    },
    {
        text: 'Tools',
        items: [
            { text: 'Spelling & Grammar' },
            { text: 'Customize' },
            { text: 'Options' }
        ]
    },
    { text: 'Go' },
    { text: 'Help' }
];
// Initialize Menu component.
let menuObj: Menu = new Menu({ items: menuItems }, '#menu');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <ul id="menu"></ul>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Navigation

Navigation in Menu is used to navigate to the other web page when a menu item is clicked. It can be achieved by providing a link to the menu item using the [url](#) property. In the following sample, the Navigation URL is added to sub menu items using the [url](#) property.

INDEX.TS

```

import { Menu, MenuItemModel, MenuEventArgs } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition
let menuItems: MenuItemModel[] = [
  {
    text: 'Appliances',

```

```

        items: [
            { text: 'Washing Machine', url:
'https://www.google.com/search?q=washing+machine' },
            { text: 'Air Conditioners', url:
'https://www.google.com/search?q=air+conditioners' }
        ]
    },
    {
        text: 'Mobile',
        items: [
            { text: 'Headphones', url:
'https://www.google.com/search?q=headphones' },
            { text: 'Memory Cards', url:
'https://www.google.com/search?q=memory+cards' },
            { text: 'Power Banks', url:
'https://www.google.com/search?q=power+banks' }
        ]
    },
    {
        text: 'Entertainment',
        items: [
            { text: 'Televisions', url:
'https://www.google.com/search?q=televisions' },
            { text: 'Home Theatres', url:
'https://www.google.com/search?q=home+theatres' },
            { text: 'Gaming Laptops', url:
'https://www.google.com/search?q=gaming+laptops' }
        ]
    },
    { text: 'Fashion', url: 'https://www.google.com/search?q=fashion' },
    { text: 'Offers', url: 'https://www.google.com/search?q=offers' }
];
//Initialize Menu component.
let menuObj: Menu = new Menu({
    items: menuItems,
    // To open url in blank page.
    beforeItemRender: (args: MenuEventArgs) => {
        if (args.item.url) {
            args.element.getElementsByTagName('a')[0].setAttribute('target',
'_blank');
        }
    }
}, '#menu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <ul id="menu"></ul>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multilevel nesting

The Menu supports multiple level nesting, and it can be achieved by mapping the [items](#) property inside the parent [menuitems](#). In the following sample, three-level nesting of menu has been provided.

INDEX.TS

```

import { Menu, MenuItemModel } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition
let menuItems: MenuItemModel[] = [
    {
        text: 'Fashion',
        items: [
            {
                text: 'Men Fashion',
                items: [
                    {
                        text: 'Personal Care',
                        items: [
                            { text: 'Trimmers' },
                            { text: 'Shavers' }
                        ]
                    }
                ]
            },
            {
                text: 'Clothing',

```



```

        items: [
            { text: 'Shirts' },
            { text: 'Jackets' },
            { text: 'Track Suits' }
        ]
    },
    { text: 'Footwear' }
]
},
{
    text: 'Women Fashion',
    items: [
        {
            text: 'Clothing',
            items: [
                { text: 'Kurtas' },
                { text: 'Salwars' },
                { text: 'Sarees' }
            ]
        },
        {
            text: 'Jewellery',
            items: [
                { text: 'Nosepins' },
                { text: 'Anklets' }
            ]
        }
    ]
}
]
},
{
    text: 'Home & Living',
    items: [
        {
            text: 'Washing Machine',
            items: [
                { text: 'Fully Automatic' },
                { text: 'Semi Automatic' }
            ]
        },
        {
            text: 'Air Conditioners',
            items: [
                { text: 'Inverter ACs' },
                { text: 'Split ACs' }
            ]
        }
    ]
},
{ text: 'Accessories' },
{ text: 'Sports' },
{ text: 'Gaming' }
];
//Initialize Menu component.
let menuObj: Menu = new Menu({ items: menuItems }, '#menu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <ul id="menu"></ul>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can achieve multi level nesting with data source by mapping **name** of the child items to the [children](#) sub-property of [fields](#) property. Also, we can specify [id](#) property for menu items. For more information, refer to the [data source binding](#) section. To open sub menu items only on item click, [showItemOnClick](#) should be set as **true**.

The below table represents the MenuItem properties and it's description.

Property Name	Type	Description
iconCss	string	Defines class/multiple classes separated by a space for the menu Item that is used to include an icon. Menu Item can include font icon and sprite image.
id	string	Specifies the id for menu item.

|separator|boolean|Specifies separator between the menu items. Separator are either horizontal or vertical lines used to group menu items.

|items|MenuItemModel[]|Specifies the sub menu items that is the array of MenuItem model/

|text|string|Specifies text for menu item.

|url|string|Specifies url for menu item that creates the anchor link to navigate to the url provided.

See Also

- [Customize menu items](#)
- [Group menu items with separator](#)

Data source binding and custom menu items in EJ2 JavaScript Menu control

Data binding

The Menu supports data source bindings such as array of JavaScript objects that can be structured as either **hierarchical** or **self-referential** data.

Hierarchical data

The Menu can be populated with hierarchical data source by assigning it to the [items](#) property, and the fields with corresponding keys can be mapped to the [fields](#) property.

JSON data

The Menu can generate its menu items through an array of complex data source by mapping fields from the [fields](#) property.

INDEX.TS

```
import { Menu, FieldSettingsModel } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
//Import an array of JSON data from datasource.ts
import { dataSource } from './datasource.ts';
enableRipple(true);
//Menu fields definition
let menuFields: FieldSettingsModel = {
    text: ['continent', 'country', 'language'],
    children: ['countries', 'languages']
};
//Initialize Menu component.
let menuObj: Menu = new Menu({ items: dataSource, fields: menuFields },
'#menu');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <ul id="menu"></ul>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Data Service

In application level, remote data binding can be achieved using [DataManager](#). To create Menu, assign `items` property with resultant data from [callback](#) function.

The following example displays five employees' **FirstName** from **Employees** table and **ShipName** details from **Orders** table of the **Northwind** Data Service.

INDEX.TS

```

import { Menu, FieldSettingsModel } from '@syncfusion/ej2-navigations';
import { DataManager, Query, ODataV4Adaptor, ReturnOption } from
 '@syncfusion/ej2-data';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
const SERVICE_URI: string =
 'https://services.odata.org/V4/Northwind/Northwind.svc/';
// Menu fields definition.
let menuFields: FieldSettingsModel = {
    text: ['FirstName', 'ShipName'],
    children: ['Orders']
};
// Getting remote data using DataManager.
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor(),
crossDomain: true })
.executeQuery(
new Query().from('Employees').take(5).hierarchy(

```

```

    new Query()
    .foreignKey('EmployeeID')
    .from('Orders').take(13),
    function() {
        return [1, 2, 3, 4, 5]
    }
))
.then((e: ReturnOption) => {
    //Initialize Menu component.
    new Menu({ items: ((Object[]>e.result) as { [key: string]: Object }[]),
fields: menuFields }, '#menu');
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <ul id="menu"></ul>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Self-referential data

Menu can be populated from self-referential data structure that contains array of JSON objects with **parentId** mapping.

In the following example, the **id**, **pId**, and **text** columns from self-referential data have been mapped to the **itemId**, **parentId**, and **text** fields, respectively.

INDEX.TS

```
import { Menu, FieldSettingsModel } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu datasource
let data: { [key: string]: Object }[] = [
  { id: 'parent1', text: 'Events' },
  { id: 'parent2', text: 'Movies' },
  { id: 'parent3', text: 'Directory' },
  { id: 'parent4', text: 'Queries', pId: null },
  { id: 'parent5', text: 'Services', pId: null },
  { id: 'parent6', text: 'Conferences', pId: 'parent1' },
  { id: 'parent7', text: 'Music', pId: 'parent1' },
  { id: 'parent8', text: 'Workshops', pId: 'parent1' },
  { id: 'parent9', text: 'Now Showing', pId: 'parent2' },
  { id: 'parent10', text: 'Coming Soon', pId: 'parent2' },
  { id: 'parent10', text: 'Media Gallery', pId: 'parent3' },
  { id: 'parent11', text: 'Newsletters', pId: 'parent3' },
  { id: 'parent12', text: 'Our Policy', pId: 'parent4' },
  { id: 'parent13', text: 'Site Map', pId: 'parent4' },
  { id: 'parent14', text: 'Pop', pId: 'parent7' },
  { id: 'parent15', text: 'Folk', pId: 'parent7' },
  { id: 'parent16', text: 'Classical', pId: 'parent7' }
];
//Menu fields definition
let menuFields: FieldSettingsModel = {
  itemId: 'id',
  text: 'text',
  parentId: 'pId'
};
//Initialize Menu component
let menuObj: Menu = new Menu({ items: data, fields: menuFields }, '#menu');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
```

```

<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <ul id="menu"></ul>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

HTML element

The Menu can be initialized using `` element that contains a collection of `` elements. A `` item acts as a menu item of the Menu, and the sub `` element inside the `` element acts as a sub menu item of its preceding menu item.

INDEX.TS

```

import { Menu } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Initialize Menu component.
let menuObj: Menu = new Menu(null, '#menu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->

```

```

<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <ul id="menu">
            <li id="leafy_salad"><a>Leafy</a>
                <ul>
                    <li>Broccoli</li>
                    <li>Cabbage</li>
                    <li>Spinach</li>
                </ul>
            </li>
            <li id="beans"><a>Beans</a>
                <ul>
                    <li>Chickpea</li>
                    <li>Green bean</li>
                    <li>Horse gram</li>
                </ul>
            </li>
            <li id="bulb_stem"><a>Bulb and Stem</a>
                <ul>
                    <li>Pearl onion</li>
                    <li>Garlic</li>
                    <li>Shallot</li>
                    <li>Lotus root</li>
                </ul>
            </li>
            <li id="root_tuberous"><a>Root</a>
                <ul>
                    <li>Beetroot</li>
                    <li>Carrot</li>
                    <li>Ginger</li>
                    <li>Turmeric</li>
                </ul>
            </li>
            <li id="pod"><a>Podded</a>
                <ul>
                    <li>American groundnut</li>
                    <li>Drumstick</li>
                    <li>Horse gram</li>
                    <li>Peanut</li>
                </ul>
            </li>
        </ul>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```



```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom menu items

The Menu can be customized using Essential JS2 [Template engine](#) to render the elements.

To customize menu items in your application, set your customized template string to the [template](#) property. In the following example, the menu has been rendered with customized menu items.

INDEX.TS

```

import { Menu, FieldSettingsModel } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(false);
//Menu data definition
let data: { [key: string]: Object }[] = [
    {
        category: 'Products',
        options: [
            { value: 'JavaScript', url: 'javascript' },
            { value: 'Angular', url: 'angular' },
            { value: 'ASP.NET Core', url: 'core' },
            { value: 'ASP.NET MVC', url: 'mvc' }
        ]
    },
    {
        category: 'Services',
        options: [
            {
                support: [
                    { value: 'Application Development', count: '1200+' },
                    { value: 'Maintenance & Support', count: '3700+' },
                    { value: 'Quality Assurance' },
                    { value: 'Cloud Integration', count: '900+' }
                ]
            }
        ]
    },
    {
        category: 'About Us',
        options: [
            {
                about: {
                    value: "We are on a mission to provide world-class best software solutions for web, mobile and desktop platforms. Around 900+ applications are desgined and delivered to our customers to make digital & strengthen their businesses."
                }
            }
        ]
    },
    { category: 'Careers' },
    { category: 'Sign In' }
];

```

```
//Menu fields definition
let menuFields: FieldSettingsModel = {
  text: ['category', 'value'],
  children: ['options']
};
//Menu initialization
let menuObj: Menu = new Menu({
  items: data,
  fields: menuFields,
  template: '#menuTemplate'
}, '#menu');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
cards/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="menu-section">
      <ul id="menu"></ul>
    </div>
  </div>
```

```

<script id="menuTemplate" type="text/x-template">
  ${if(category)}
    ${category}
  ${else if (value && url)}
    <div class="e-avatar e-avatar-small image" style="background-
image: url(${url}.png);">${value}</div>
  ${else if (support)}
    <ul>
      ${for(val of support)}
      <li>
        ${val.value}
        ${if(val.count)}
          <span class='e-badge e-badge-
success'>${val.count}</span>
        ${/if}
      </li>
      ${/for}
    </ul>
  ${else}
    <div tabindex="0" class="e-card">
      <div class="e-card-header">
        <div class="e-card-header-caption">
          <div class="e-card-header-title">About Us</div>
        </div>
      </div>
      <div class="e-card-content">
        ${about.value}
      </div>
      <div class="e-card-actions">
        <button class="e-btn e-outline">
          Read More
        </button>
      </div>
    </div>
  ${/if}
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To prevent sub menu closing, set `args.cancel` to `true` in [beforeClose](#) event.

See Also

- [Render menu with items](#)

Use case scenarios in EJ2 JavaScript Menu control

Scrollable menu

The menu component supports horizontal and vertical scrolling to render large menus and submenus in an adaptive way. This can be achieved by enabling the [enableScrolling](#) property and by restricting the corresponding menu/submenu size.

INDEX.TS

```
import { Menu, MenuItemModel, BeforeOpenCloseMenuEventArgs, MenuModel } from
 '@syncfusion/ej2-navigations';
import { enableRipple, closest } from '@syncfusion/ej2-base';
enableRipple(true);
// Menu items definition
let menuItems: MenuItemModel[] = [
    {
        text: 'Appliances',
        items: [
            {
                text: 'Kitchen',
                items: [
                    { text: 'Electric Cookers' },
                    { text: 'Coffee Makers' },
                    { text: 'Blenders' },
                    { text: 'Microwave Ovens' }
                ]
            },
            {
                text: 'Television',
                items: [
                    { text: 'Our Exclusive TVs' },
                    { text: 'Smart TVs' },
                    { text: 'Big Screen TVs' }
                ]
            },
            {
                text: 'Washing Machine'
            },
            {
                text: 'Refrigerators'
            },
            {
                text: 'Air Conditioners',
                items: [
                    { text: 'Inverter ACs' },
                    { text: 'Split ACs' },
                    { text: 'Window ACs' }
                ]
            },
            {
                text: 'Water Purifiers'
            },
            {
                text: 'Air Purifiers'
            },
            {
                text: 'Chimneys'
            }
        ]
    }
];
```

```

        },
        {
            text: 'Inverters'
        },
        {
            text: 'Healthy Living'
        },
        {
            text: 'Vacuum Cleaners'
        },
        {
            text: 'Room Heaters'
        },
        {
            text: 'New Launches'
        }
    ]
},
{
    text: 'Accessories',
    items: [
        {
            text: 'Mobile',
            items: [
                { text: 'Headphones' },
                { text: 'Memory Cards' },
                { text: 'Power Banks' },
                { text: 'Mobile Cases' },
                { text: 'Screen Protectors' }
            ]
        },
        {
            text: 'Laptops'
        },
        {
            text: 'Desktop PC',
            items: [
                { text: 'Pendrives' },
                { text: 'External Hard Disks' },
                { text: 'Monitors' },
                { text: 'Keyboards' }
            ]
        },
        {
            text: 'Camera',
            items: [
                { text: 'Lens' },
                { text: 'Tripods' }
            ]
        }
    ]
},
{
    text: 'Fashion',
    items: [
        {
            text: 'Men'
        }
    ]
}

```

```

        },
        {
            text: 'Women'
        }
    ]
},
{
    text: 'Home & Living',
    items: [
        {
            text: 'Furniture'
        },
        {
            text: 'Decor'
        },
        {
            text: 'Smart Home Automation'
        },
        {
            text: 'Dining & Serving'
        }
    ]
},
{
    text: 'Entertainment',
    items: [
        {
            text: 'Televisions'
        },
        {
            text: 'Home Theatres'
        },
        {
            text: 'Gaming Laptops'
        }
    ]
},
{
    text: 'Contact Us'
},
{
    text: 'Help'
}
];
let menuOptions: MenuModel = {
    items: menuItems,
    cssClass: 'e-scrollable-menu',
    enableScrolling: true,
    beforeOpen: (args: BeforeOpenCloseMenuEventArgs): void => {
        // Restricting sub menu wrapper height
        if (args.parentItem.text === 'Appliances') {
            (closest(args.element, '.e-menu-wrapper') as
HTMLInputElement).style.height = '230px';
        }
    }
};
new Menu(menuOptions, '#menu');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <ul id="menu"></ul>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Menu in toolbar

The following example demonstrates how to integrate Menu with Toolbar component.

INDEX.TS

```

import { Toolbar, Menu, MenuItemModel } from '@syncfusion/ej2-navigations';
let menuTemplate: string = '<ul id="menu"></ul>';
//Initialize Toolbar component
let toolbarObj: Toolbar = new Toolbar({
  created: create,
  items: [
    { template: menuTemplate },
    { prefixIcon: 'em-icons e-shopping-cart', align: 'Right' }
  ]
}

```

```

});
//Render initialized Toolbar component
toolbarObj.appendTo('#shoppingtoolbar');
function create(): void {
    let data: { [key: string]: Object }[] = [
        {
            header: 'Events',
            subItems: [
                { text: 'Conferences' },
                { text: 'Music' },
                { text: 'Workshops' }
            ]
        },
        {
            header: 'Movies',
            subItems: [
                { text: 'Now Showing' },
                { text: 'Coming Soon' }
            ]
        },
        {
            header: 'Directory',
            subItems: [
                { text: 'Media Gallery' },
                { text: 'Newsletters' }
            ]
        },
        {
            header: 'Queries',
            subItems: [
                { text: 'Our Policy' },
                { text: 'Site Map' },
                { text: '24x7 Support' }
            ]
        },
        { header: 'Services' }
    ];
    //Menu fields definition
    let menuFields: Object = {
        text: ['header', 'text', 'value'],
        children: ['subItems', 'options']
    };
    //Initialize Menu component
    let menuObj: Menu = new Menu({
        items: data,
        fields: menuFields,
        animationSettings: { effect: 'None' }
    }, '#menu');
    this.refreshOverflow();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2</title>
<meta charset="utf-8">

```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="toolbar-menu-control">
                <div id="shoppingtoolbar"></div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hamburger menu

The following example demonstrates the use case of menu with Accordion component integrated in SideBar.

INDEX.TS

```

import { Sidebar, Accordion, ExpandEventArgs, AccordionClickArgs } from
'@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let acrdnObj: Accordion;
// Sidebar Initialization
let defaultSidebar: Sidebar = new Sidebar({ width: '220px', type: 'Over',
created: created });
defaultSidebar.appendTo('#default-sidebar');
// Close the Sidebar
document.getElementById('close').onclick = (): void => {
    defaultSidebar.hide();
};

```

```

};
document.getElementById('hamburger').onclick = (): void => {
    defaultSidebar.show();
    acrdnObj.refresh();
};
function created(): void {
    let data: { [key: string]: Object }[] = [
        {
            header: 'Appliances',
            content: '<div id="Appliances_Items"></div>',
            subItems: [
                {
                    header: 'Kitchen',
                    content: '<div id="Appliances_Kitchen_Items"></div>',
                    subItems: [
                        { header: 'Electric Cookers' },
                        { header: 'Coffee Makers' },
                        { header: 'Blenders' },
                    ]
                },
                {
                    header: 'Washing Machine',
                    content: '<div id="Appliances_Washing_Items"></div>',
                    subItems: [
                        { header: 'Fully Automatic' },
                        { header: 'Semi Automatic' }
                    ]
                },
                {
                    header: 'Air Conditioners',
                    content: '<div id="Appliances_Conditioners_Items"></div>',
                    subItems: [
                        { header: 'Inverter ACs' },
                        { header: 'Split ACs' },
                        { header: 'Window ACs' },
                    ]
                }
            ]
        },
        {
            header: 'Accessories',
            content: '<div id="Accessories_Items"></div>',
            subItems: [
                {
                    header: 'Mobile',
                    content: '<div id="Accessories_Mobile_Items"></div>',
                    subItems: [
                        { header: 'Headphones' },
                        { header: 'Memory Cards' },
                        { header: 'Power Banks' }
                    ]
                },
                {
                    header: 'Computer',
                    content: '<div id="Accessories_Computer_Items"></div>',
                    subItems: [

```

```

        { header: 'Pendrives' },
        { header: 'External Hard Disks' },
        { header: 'Monitors' }
    ]
}
],
{
    header: 'Fashion',
    content: '<div id="Fashion_Items"></div>',
    subItems: [
        { header: 'Men' },
        { header: 'Women' }
    ]
},
{
    header: 'Home & Living',
    content: '<div id="Home_Living_Items"></div>',
    subItems: [
        { header: 'Furniture' },
        { header: 'Decor' }
    ]
},
{
    header: 'Entertainment',
    content: '<div id="Entertainment_Items"></div>',
    subItems: [
        { header: 'Televisions' },
        { header: 'Home Theatres' },
        { header: 'Gaming Laptops' }
    ]
}
];
//Initialize Accordion component
acrdnObj = new Accordion ({
    items: data,
    expanding: expand,
    clicked: clicked
}, '#accordionMenu');
}
//Expanding Event function for Accordion component.
function expand(e: ExpandEventArgs): void {
    if (e.isExpanded) {
        if (e.element.getElementsByClassName('e-acrdn-content')[0].children[0].classList.contains('e-accordion')) {
            return;
        }
        //Initialize Nested Accordion component
        let nestAcrdn: Accordion = new Accordion({
            items: (<{ subItems: object[] }>e.item).subItems,
            expanding: expand,
            clicked: clicked
        });
        let elemId: string = e.element.getElementsByClassName('e-acrdn-content')[0].children[0].id;
        //Render initialized Nested Accordion component
        nestAcrdn.appendTo('#' + elemId);
    }
}

```

```

    }
}
function clicked(e: AccordionClickArgs): void {
    if (!e.item && !(e.originalEvent.target as HTMLElement).closest('.e-acrdn-item').getElementsByClassName('e-tgl-collapse-icon').length) {
        defaultSidebar.hide();
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- sidebar element declaration -->
        <div class="header">
            <span id="hamburger" class="e-icons menu default"></span>
            <div class="content">Header content</div>
        </div>
        <aside id="default-sidebar">
            <div class="title-header">
                <div style="display:inline-block"> Menu </div>
                <span id="close" class="e-icons"></span>
            </div>
            <div class="content-area">
                <!--Accordion control declaration-->
                <div id="accordionMenu"></div>
            </div>
        </aside>
        <!-- main content declaration -->
        <div>
            <div class="main-content">Main content</div>
        </div>
    </div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Mobile view

The following example demonstrates the use case of Menu in Mobile mode by using ListView component with hamburger.

INDEX.TS

```

import { enableRipple, AnimationOptions } from '@syncfusion/ej2-base';
import { ListView, SelectEventArgs } from '@syncfusion/ej2-lists';
import { Animation } from '@syncfusion/ej2-base';
enableRipple(true);
let dataSource: { [key: string]: Object }[] = [
    {
        text: 'Appliances',
        id: 'list1',
        child: [
            {
                text: 'Kitchen',
                id: 'list1_1',
                child: [
                    { id: 'list1_1_1', text: 'Electric Cookers' },
                    { id: 'list1_1_2', text: 'Coffee Makers' },
                    { id: 'list1_1_3', text: 'Blenders' },
                ]
            },
            {
                text: 'Washing Machine',
                id: 'list1_2',
                child: [
                    { id: 'list1_2_1', text: 'Fully Automatic' },
                    { id: 'list1_2_2', text: 'Semi Automatic' }
                ]
            },
            {
                text: 'Air Conditioners',
                id: 'list1_3',
                child: [
                    { id: 'list1_3_1', text: 'Inverter ACs' },
                    { id: 'list1_3_2', text: 'Split ACs' },
                    { id: 'list1_3_3', text: 'Window ACs' },
                ]
            }
        ]
    },
    {
        text: 'Accessories',
        id: 'list2',
    }
]

```

```

        child: [
            {
                text: 'Mobile',
                id: 'list2_1',
                child: [
                    { id: 'list2_1_1', text: 'Headphones' },
                    { id: 'list2_1_2', text: 'Memory Cards' },
                    { id: 'list2_1_3', text: 'Power Banks' }
                ]
            },
            {
                text: 'Computer',
                id: 'list2_2',
                child: [
                    { id: 'list2_2_1', text: 'Pendrives' },
                    { id: 'list2_2_2', text: 'External Hard Disks' },
                    { id: 'list2_2_3', text: 'Monitors' }
                ]
            }
        ],
        {
            text: 'Fashion',
            id: 'list3',
            child: [
                { id: 'list3_1', text: 'Men' },
                { id: 'list3_2', text: 'Women' }
            ]
        },
        {
            text: 'Home & Living',
            id: 'list4',
            child: [
                { id: 'list4_1', text: 'Furniture' },
                { id: 'list4_2', text: 'Decor' }
            ]
        },
        {
            text: 'Entertainment',
            id: 'list5',
            child: [
                { id: 'list5_1', text: 'Televisions' },
                { id: 'list5_2', text: 'Home Theatres' },
                { id: 'list5_3', text: 'Gaming Laptops' }
            ]
        }
    ];
    //Initialize ListView component
    let listObj: ListView = new ListView({
        //Set defined data to dataSource property
        dataSource: dataSource,
        //Set header title
        headerTitle: 'Menu',
        //Set true to show header title
        showHeader: true,
        select: onSelect
    });

```

```
//Render initialized ListView component
listObj.appendTo('#listview');
document.getElementById('hamburger').onclick = (): void => {
    let animation: Animation = new Animation({ duration: 500 });
    animation.animate(listObj.element, {
        name: 'SlideDown' ,
        begin: function(args: AnimationOptions) {
            listObj.element.style.display = 'block';
            document.getElementById('close').style.display = 'block';
        }
    });
};
// Close the ListView
document.getElementById('close').onclick = (): void => {
    listObj.element.style.display = 'none';
    document.getElementById('close').style.display = 'none';
};
function onSelect (e: SelectEventArgs): void {
    if (e.data && !(e.data as { child: object }).child) {
        listObj.element.style.display = 'none';
        document.getElementById('close').style.display = 'none';
        listObj.refresh();
    }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="layoutWrapper">
            <div class="speaker">
                <div class="camera"></div>
```

```

        </div>
        <div class="layout">
            <div id="container">
                <div id="header">
                    <span id="hamburger" class="e-icons menu
default"></span>
                    <div class="content">Header</div>
                </div>
                <!-- ListView element -->
                <div id="listview" tabindex="1" style="display:
none;"></div>
                <span id="close" class="e-icons" style="display:
none;"></span>
            </div>
        </div>
        <div class="outerButton"> </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in EJ2 JavaScript Menu control

The Menu component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Menu component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

```
<style>
```

```
.post .post-content img {
```

```
display: inline-block;
```

```
margin: 0.5em 0;
```

```
}
```

```
</style>
```

```
<div> - All features of the component meet the requirement.</div>
```

```
<div> - Some features of the component do not meet the requirement.</div>
```

```
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Menu component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Menu component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates Menu component's root menu as **menubar**, popup as **menu**, and the popup items as **menuitem**. |

| **aria-haspopup** | Indicates the availability and type of interactive popup element. |

| **aria-expanded** | Indicates whether the subtree can be expanded or collapsed, and indicates whether its current state can be expanded or collapsed. |

| **aria-orientation** | Indicates whether the orientation is horizontal or vertical. The default orientation is horizontal. |

| **aria-label** | Indicates the menu item text. |

| **aria-disabled** | Indicates the state of menu item whether it is disabled. |

Keyboard interaction

The Menu component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Menu component.

| **Press** | **To do this** |

| --- | --- |

| **Esc** | Closes the sub menu that contains focus and returns focus to the parent element. |

| Enter | Opens the sub menu if focused menu item has sub menu, and places focus on its first item or activates the item and closes the sub menu. |

| Up | Navigates up or to the previous menu item. |

| Down | Navigates down or to the next menu item. |

| Left | Closes the current sub menu and navigates to the parent menu. |

| Right | Navigates and open the next sub menu. |

| Home | Focuses the first item. |

| End | Focuses the last item.

Ensuring accessibility

The Menu component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Menu component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Menu component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

Style and appearance in EJ2 JavaScript Menu control

To modify the Menu appearance, you need to override the default CSS of Menu component. Please find the list of CSS classes and its corresponding section in Menu component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class | Purpose of Class

|.e-menu-wrapper|To customize the menu wrapper

|.e-menu-wrapper.e-menu-popup|To customize the menu popup wrapper

|.e-menu-wrapper ul .e-menu-item|To customize the menu items

|.e-menu-wrapper.e-menu-popup .e-menu-item|To customize the menu popup items

|.e-menu-wrapper ul .e-menu-item .e-caret|To customize the menu items caret icon

How To

Change orientation in EJ2 JavaScript Menu control

Orientation in menu items can be changed horizontally or vertically using the [orientation](#) property. By default, it is horizontally aligned.

INDEX.TS

```
import { Menu, MenuItemModel } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition
let menuItems: MenuItemModel[] = [
  {
    text: 'File',
```

```

        items: [
            { text: 'Open' },
            { text: 'Save' },
            { text: 'Exit' }
        ]
    },
    {
        text: 'Edit',
        items: [
            { text: 'Cut' },
            { text: 'Copy' },
            { text: 'Paste' }
        ]
    },
    {
        text: 'View',
        items: [
            { text: 'Toolbar' },
            { text: 'Sidebar' },
            { text: 'Full Screen' }
        ]
    },
    {
        text: 'Tools',
        items: [
            { text: 'Spelling & Grammar' },
            { text: 'Customize' },
            { text: 'Options' }
        ]
    },
    { text: 'Help' }
];
// Initialize Menu component.
let menuObj: Menu = new Menu({ orientation: 'Vertical', items: menuItems },
'#menu');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <ul id="menu"></ul>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize menu using css in EJ2 JavaScript Menu control

Menu provides a set of css class to enable users to customize it. The following list of CSS class names are used to customize the Menu component.

You can customize the appearance of the Menu by overriding the existing styles. The following list of CSS class names are used to customize the menu component.

Class Name	Description
---	---
e-menu-wrapper	Applied to Menu wrapper element.
e-menu-parent e-menu	Applied to the parent item element.
e-menu-item	Applied to the menu item.
e-menu-icon	Applied to the span element of menu item.
e-menu-url	Applied to the URL menu item.
e-menu-parent e-ul	Applied to the sub menu's parent ul element.
e-menu-popup e-popup	Applied to the element when menu item popups.
e-selected	Applied to the selected menu item.
e-focused	Applied to the focused menu item.
e-menu-caret-icon	Applied to the menu item with caret icon.
e-hamburger	Applied to the element with hamburger mode.
e-disabled	Applied to the disabled menu element.
e-vertical	Applied to the vertically oriented menu element.
e-rtl	Applied to the menu with rtl state.

- | e-bigger | Applied to the bigger menu.
- | e-blankicon | Applied to the menu items with no icons.
- | e-scroll-right-nav | Applied to the element with right scroll .
- | e-scroll-left-nav | Applied to the element with left scroll .

Customize menu using [events](#) in EJ2 JavaScript Menu control

The Menu provides a set of [events](#) to enable users to customize it.

The available events are [beforeOpen](#), [beforeClose](#), [onClose](#), [onOpen](#), and [select](#).

INDEX.TS

```
import { Menu, MenuItemModel, MenuEventArgs } from '@syncfusion/ej2-
navigations';
import { OpenCloseMenuEventArgs, BeforeOpenCloseMenuEventArgs } from
 '@syncfusion/ej2-navigations';
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition
let menuItems: MenuItemModel[] = [
    {
        text: 'Events',
        items: [
            { text: 'Conferences' },
            { text: 'Music' },
            { text: 'Workshops' }
        ]
    },
    {
        text: 'Movies',
        items: [
            { text: 'Now Showing' },
            { text: 'Coming Soon' }
        ]
    },
    {
        text: 'Directory',
        items: [
            { text: 'Media Gallery' },
            { text: 'Newsletters' }
        ]
    },
    {
        text: 'Queries',
        items: [
            { text: 'Our Policy' },
            { text: 'Site Map' }
        ]
    },
    { text: 'Services' }
];
// Initialize Menu component.
let menuObj: Menu = new Menu({
    items: menuItems,
```

```

beforeOpen: (args: BeforeOpenCloseMenuEventArgs) => {
    updateEventLog(args);
},
beforeClose: (args: BeforeOpenCloseMenuEventArgs) => {
    updateEventLog(args);
},
onClose: (args: OpenCloseMenuEventArgs) => {
    updateEventLog(args);
},
onOpen: (args: OpenCloseMenuEventArgs) => {
    updateEventLog(args);
},
select: (args: MenuEventArgs) => {
    updateEventLog(args);
}
}, '#menu');
let clear: Button = new Button({ cssClass: 'e-small' });
clear.appendTo('#clear');
clear.element.onclick = () => {
    let propertyElem: HTMLElement =
document.getElementById('propertyTable');
    propertyElem.getElementsByTagName('td')[0].innerHTML = '';
}
function updateEventLog(args: any): void {
    let propertyElem: HTMLElement =
document.getElementById('propertyTable');

propertyElem.getElementsByTagName('td')[0].insertAdjacentHTML('beforeend',
args.name + ' Event triggered. <br />');
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="menu-section">
                <ul id="menu"></ul>
            </div>
            <div class="property-section">
                <table id="propertyTable" title="Event trace">
                    <tbody>
                        <tr><th>Event trace:-</th>
                        </tr><tr>
                            <td></td>
                        </tr>
                    </tbody>
                </table>
            </div>
            <button id="clear">Clear</button>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize menu items in EJ2 JavaScript Menu control

Add or remove menu items

Menu items can be added or removed using the [insertAfter](#), [insertBefore](#) and [removeItems](#) methods.

In the following example, the **Europe** menu items are added before the **Oceania** item, the **Africa** menu items are added after the **Asia**, and the **South America** and **Mexico** items are removed from menu.

INDEX.TS

```

import { Menu, FieldSettingsModel } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu datasource
let data: { [key: string]: Object }[] = [
    {
        continent: 'Asia',
        countries: [
            { country: 'China' },
            { country: 'India' },
            { country: 'Japan' }
        ]
    },
    {
        continent: 'North America',

```

```

        countries: [
            { country: 'Canada' },
            { country: 'Mexico' },
            { country: 'USA' }
        ]
    },
    {
        continent: 'South America',
        countries: [
            { country: 'Brazil' },
            { country: 'Colombia' },
            { country: 'Argentina' }
        ]
    },
    {
        continent: 'Oceania',
        countries: [
            { country: 'Australia' },
            { country: 'New Zealand' },
            { country: 'Samoa' },
        ]
    },
    { continent: 'Antarctica' }
];
//Menu fields definition
let menuFields: FieldSettingsModel = {
    text: ['continent', 'country'],
    children: ['countries']
};
//Initialize Menu component.
let menuObj: Menu = new Menu({ items: data, fields: menuFields }, '#menu');
let insertItem: object[] = [
    {
        continent: 'Europe',
        countries: [
            { country: 'Finland' },
            { country: 'Austria' }
        ]
    }
];
//Add items before to 'Oceania'
menuObj.insertBefore(insertItem, 'Oceania', false);
insertItem = [
    {
        continent: 'Africa',
        countries: [
            { country: 'Nigeria' }
        ]
    }
];
//Add items after to 'Asia'
menuObj.insertAfter(insertItem, 'Asia', false);
//Remove items
menuObj.removeItem(['South America', 'Mexico'], false);

```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <ul id="menu"></ul>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To process items with ID values, set `isUnique` to `true`.

Enable or disable menu items

You can enable and disable the menu items using the [enableItems](#) method in Menu. To enable menuItems, set the `enable` property in argument to `true` and vice-versa.

In the following example, the **Directory** header item, **Conferences**, and **Music** sub menu items are disabled.

INDEX.TS

```

import { Menu, MenuItemModel, BeforeOpenCloseMenuEventArgs } from
'@syncfusion/ej2-navigations';
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition

```

```

let menuItems: MenuItemModel[] = [
    {
        text: 'Events',
        items: [
            { text: 'Conferences' },
            { text: 'Music' },
            { text: 'Workshops' }
        ]
    },
    {
        text: 'Movies',
        items: [
            { text: 'Now Showing' },
            { text: 'Coming Soon' }
        ]
    },
    {
        text: 'Directory',
        items: [
            { text: 'Media Gallery' },
            { text: 'Newsletters' }
        ]
    },
    {
        text: 'Queries',
        items: [
            { text: 'Our Policy' },
            { text: 'Site Map' }
        ]
    },
    { text: 'Services' }
];
// Initialize Menu component.
let menuObj: Menu = new Menu({
    items: menuItems,
    beforeOpen: (args: BeforeOpenCloseMenuEventArgs) => {
        //Handling sub menu items
        for (let i: number = 0; i < args.items.length; i++) {
            if (disableItems.indexOf(args.items[i].text) > -1) {
                menuObj.enableItems([args.items[i].text], false, false);
            }
        }
    },
    '#menu');
let disableItems: string[] = ['Conferences', 'Music', 'Directory'];
//Disable items
menuObj.enableItems(disableItems, false, false);
let buttonObj: Button = new Button();
buttonObj.appendTo('#enableAll');
buttonObj.element.onclick = (): void => {
    menuObj.enableItems(disableItems, true, false);
    disableItems = [];
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <button id="enableAll">Enable all items</button>
      <div class="menu-section">
        <ul id="menu"></ul>
      </div>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To disable sub menu items, use the [beforeOpen](#) event.

Hide or show menu items

You can show or hide the menu items using the [showItems](#) and [hideItems](#) methods.

In the following example, the **Movies** header item, **Workshops**, and **Music** sub menu items are hidden in menu.

INDEX.TS

```

import { Menu, MenuItemModel, BeforeOpenCloseMenuEventArgs } from
'@syncfusion/ej2-navigations';

```

```

import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition
let menuItems: MenuItemModel[] = [
    {
        text: 'Events',
        items: [
            { text: 'Conferences' },
            { text: 'Music' },
            { text: 'Workshops' }
        ]
    },
    {
        text: 'Movies',
        items: [
            { text: 'Now Showing' },
            { text: 'Coming Soon' }
        ]
    },
    {
        text: 'Directory',
        items: [
            { text: 'Media Gallery' },
            { text: 'Newsletters' }
        ]
    },
    {
        text: 'Queries',
        items: [
            { text: 'Our Policy' },
            { text: 'Site Map' }
        ]
    },
    { text: 'Services' }
];
// Initialize Menu component.
let menuObj: Menu = new Menu({
    items: menuItems,
    beforeOpen: (args: BeforeOpenCloseMenuEventArgs) => {
        //Handling sub menu items
        for (let i: number = 0; i < args.items.length; i++) {
            if (hiddenItems.indexOf(args.items[i].text) > -1) {
                menuObj.hideItems([args.items[i].text], false);
            }
        }
    },
    '#menu');
let hiddenItems: string[] = ['Workshops', 'Music', 'Movies'];
//Hide items
menuObj.hideItems(hiddenItems, false);
let buttonObj: Button = new Button();
buttonObj.appendTo('#showAll');
buttonObj.element.onclick = (): void => {
    menuObj.showItems(hiddenItems, false);
    hiddenItems = [];
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <button id="showAll">Show all items</button>
      <div class="menu-section">
        <ul id="menu"></ul>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Using the [beforeOpen](#) event, you can hide the sub menu items as in the above example since the menu supports to hide items only for headers initially.

Create mnemonic ui in menuitem in EJ2 JavaScript Menu control

A particular character in a text can be underlined in the [beforeItemRender](#) event by adding `<u>` tag in between the text and assign the innerHTML to the `li` element.

In the following example, the first character in `File`, `Open`, and `Save` menu items are underlined.

INDEX.TS

```

import { Menu, MenuItemModel, MenuEventArgs } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition
let menuItems: MenuItemModel[] = [
    {
        text: 'File',
        iconCss: 'em-icons e-file',
        items: [
            { text: 'Open', iconCss: 'em-icons e-open' },
            { text: 'Save', iconCss: 'em-icons e-save' },
            { separator: true },
            { text: 'Exit' }
        ]
    },
    {
        text: 'Edit',
        iconCss: 'em-icons e-edit',
        items: [
            { text: 'Cut', iconCss: 'em-icons e-cut' },
            { text: 'Copy', iconCss: 'em-icons e-copy' },
            { text: 'Paste', iconCss: 'em-icons e-paste' }
        ]
    },
    { text: 'Format' },
    { text: 'View' },
    { text: 'Bookmarks' },
    { text: 'Tools' },
    { separator: true },
    { text: 'Help' }
];
// Initialize Menu component.
let menuObj: Menu = new Menu({
    items: menuItems,
    beforeItemRender: (args: MenuEventArgs) => {
        if ([ 'File', 'Open', 'Save' ].indexOf(args.item.text) > -1) {
            // To underline a First character.
            let underlinedText: string = '<u>' + args.item.text.slice(0, 1)
+ '</u>' + args.item.text.slice(1);
            args.element.innerHTML =
args.element.innerHTML.replace(args.item.text, underlinedText);
        }
    }
}, '#menu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">

```

```

<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <ul id="menu"></ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change sub menu position in EJ2 JavaScript Menu control

The submenu position can be changed by using the [beforeOpen](#) event. Assign the top and left position where you want to open the submenu to the [beforeOpen](#) event arguments `args.top` and `args.left` respectively.

In the below sample, the sub menu opens above the parent menu item.

INDEX.TS

```

import { Menu, MenuItemModel, MenuModel } from '@syncfusion/ej2-
navigations';
import { BeforeOpenCloseMenuEventArgs } from '@syncfusion/ej2-navigations';
import { enableRipple, closest } from '@syncfusion/ej2-base';
enableRipple(true);
// Menu items definition
let menuItems: MenuItemModel[] = [
    {
        text: 'File',
        items: [
            { text: 'Open' },
            { text: 'Save' },
            { text: 'Exit' }
        ]
    },
    {

```

```

        text: 'Edit',
        items: [
            { text: 'Cut' },
            { text: 'Copy' },
            { text: 'Paste' }
        ]
    },
    {
        text: 'View',
        items: [
            { text: 'Toolbar' },
            { text: 'Sidebar' }
        ]
    },
    {
        text: 'Tools',
        items: [
            { text: 'Spelling & Grammar' },
            { text: 'Customize' },
            { text: 'Options' }
        ]
    },
    { text: 'Go' },
    { text: 'Help' }
];
// Menu model definitions
let menuOptions: MenuModel = {
    items: menuItems,
    beforeOpen: (args: BeforeOpenCloseMenuEventArgs) => {
        // Getting parent menu item element offset
        let relativeOffset: ClientRect = closest(args.event.target as
Element, '.e-menu-item').getBoundingClientRect();
        // Getting sub menu wrapper element using closest method
        let subMenuEle: HTMLElement = closest(args.element, '.e-menu-
wrapper') as HTMLElement;
        subMenuEle.style.display = 'block';
        args.top = (relativeOffset.top -
subMenuEle.getBoundingClientRect().height) + pageYOffset;
        args.left = relativeOffset.left + pageXOffset;
        subMenuEle.style.display = '';
    }
};
// Initialize Menu component
new Menu(menuOptions, '#menu');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <ul id="menu"></ul>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

> For custom positioning, set both **top** and **left** position in the [beforeOpen](#) event.

Rounded corner in EJ2 JavaScript Menu control

The rounded corner can be achieved by using the [cssClass](#) property. Add a custom class to the menu component and customize it using the **border-radius** CSS property. For more information, refer to the **style.css** file mapped under the source tab.

INDEX.TS

```

import { Menu, MenuItemModel } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Menu items definition
let menuItems: MenuItemModel[] = [
    {
        text: 'File',
        items: [
            { text: 'Open' },
            { text: 'Save' },
            { text: 'Exit' }
        ]
    },
    {
        text: 'Edit',
        items: [
            { text: 'Cut' },

```

```

        { text: 'Copy' },
        { text: 'Paste' }
    ],
    {
        text: 'View',
        items: [
            { text: 'Toolbar' },
            { text: 'Sidebar' }
        ]
    },
    {
        text: 'Tools',
        items: [
            { text: 'Spelling & Grammar' },
            { text: 'Customize' },
            { text: 'Options' }
        ]
    },
    { text: 'Go' },
    { text: 'Help' }
];
new Menu({ items: menuItems, cssClass: 'e-rounded-menu' }, '#menu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/bootstrap.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/bootstrap.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/bootstrap.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <ul id="menu"></ul>
        </div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Change animation settings in EJ2 JavaScript Menu control

To change the animation of the Menu, [animationSettings](#) property is used. The supported effects for Menu are,

| Effect | Functionality |

| ----- | ----- |

| None | Specifies the sub menu transform with no animation effect. |

| SlideDown | Specifies the sub menu transform with slide down effect. |

| ZoomIn | Specifies the sub menu transform with zoom in effect. |

| FadeIn | Specifies the sub menu transform with fade in effect. |

The following sample illustrates how to open Menu with **FadeIn** [effect](#) with the [duration](#) of **800ms**. Also we can set [easing](#) for menu items.

INDEX.TS

```
import { Menu, MenuItemModel, MenuModel } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition
let menuItems: MenuItemModel[] = [
    {
        text: 'Fashion',
        items: [
            {
                text: 'Men Fashion',
                items: [
                    {
                        text: 'Personal Care',
                        items: [
                            { text: 'Trimmers' },
                            { text: 'Shavers' }
                        ]
                    },
                    {
                        text: 'Clothing',
                        items: [
                            { text: 'Shirts' },
                            { text: 'Jackets' },
                            { text: 'Track Suits' }
                        ]
                    },
                    { text: 'Footwear' }
                ]
            }
        ]
    }
];
```

```

    ],
    {
      text: 'Women Fashion',
      items: [
        {
          text: 'Clothing',
          items: [
            { text: 'Kurtas' },
            { text: 'Salwars' },
            { text: 'Sarees' }
          ]
        },
        {
          text: 'Jewellery',
          items: [
            { text: 'Nosepins' },
            { text: 'Anklets' }
          ]
        }
      ]
    }
  ],
},
{
  text: 'Home & Living',
  items: [
    {
      text: 'Washing Machine',
      items: [
        { text: 'Fully Automatic' },
        { text: 'Semi Automatic' }
      ]
    },
    {
      text: 'Air Conditioners',
      items: [
        { text: 'Inverter ACs' },
        { text: 'Split ACs' }
      ]
    }
  ]
},
{ text: 'Accessories' },
{ text: 'Sports' },
{ text: 'Gaming' }
];
let menuOptions: MenuModel = {
  items: menuItems,
  animationSettings: {
    effect: 'FadeIn',
    duration: 800
  }
};
//Initialize Menu component.
let menuObj: Menu = new Menu(menuOptions, '#menu');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="control-section">
      <ul id="menu"></ul>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Right to left in EJ2 JavaScript Menu control

Menu component has RTL support. This can be achieved by setting [enableRtl](#) as `true`.

The following example illustrates how to enable right-to-left support in Menu component.

INDEX.TS

```

import { Menu, MenuItemModel } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition
let menuItems: MenuItemModel[] = [
  {
    text: 'File',

```

```

        items: [
            { text: 'Open' },
            { text: 'Save' },
            { text: 'Exit' }
        ]
    },
    {
        text: 'Edit',
        items: [
            { text: 'Cut' },
            { text: 'Copy' },
            { text: 'Paste' }
        ]
    },
    {
        text: 'View',
        items: [
            { text: 'Toolbar' },
            { text: 'Sidebar' }
        ]
    },
    {
        text: 'Tools',
        items: [
            { text: 'Spelling & Grammar' },
            { text: 'Customize' },
            { text: 'Options' }
        ]
    },
    { text: 'Go' },
    { text: 'Help' }
];
// Initialize Menu component.
let menuObj: Menu = new Menu({ items: menuItems, enableRtl: true },
'#menu');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <ul id="menu"></ul>
        </div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Set title in EJ2 JavaScript Menu control

In Menu , we can provide title for menu items by using title property in 'beforeItemRender' client-side event in Menu component.

INDEX.TS

```

import { Menu, MenuItemModel, MenuEventArgs } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let menuItems: MenuItemModel[] = [
    {
        id: 'settingIcon',
        iconCss: 'em-icons e-file',
        items:
        { text: 'Open',
            items: [
                { text: 'Sub Option1' },
                { text: 'Sub Option2' },
            ]
        },
        { text: 'Save' },
        { separator: true },
        { text: 'Exit' }
    ]
};
// Initialize Menu component.
let menuObj: Menu = new Menu({ items: menuItems ,
    beforeItemRender: (args: MenuEventArgs) => {
        if (args.item.id == 'settingIcon') {
            args.element.setAttribute('title', 'Settings')
        }
    }}, '#menu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <ul id="menu"></ul>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Menu item click in EJ2 JavaScript Menu control

You can open menu items and sub menu on menu item click by setting [showItemOnClick](#) property of the Menu. To open sub menu items only on item click, should be set as `true`.

INDEX.TS

```

import { Menu, MenuItemModel } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Menu items definition
let menuItems: MenuItemModel[] = [
  {
    text: 'File',
    items: [
      { text: 'Open' },
      { text: 'Save' },
      { text: 'Exit' }
    ]
  }
]

```



```

    ],
    {
      text: 'Edit',
      items: [
        { text: 'Cut' },
        { text: 'Copy' },
        { text: 'Paste' }
      ]
    },
    {
      text: 'View',
      items: [
        {
          text: 'Toolbars',
          items: [
            { text: 'Menu Bar' },
            { text: 'Bookmarks Toolbar' },
            { text: 'Customize' },
          ]
        },
        {
          text: 'Zoom',
          items: [
            { text: 'Zoom In' },
            { text: 'Zoom Out' },
            { text: 'Reset' },
          ]
        },
        { text: 'Full Screen' }
      ]
    },
    {
      text: 'Tools',
      items: [
        { text: 'Spelling & Grammar' },
        { text: 'Customize' },
        { text: 'Options' }
      ]
    },
    { text: 'Go' },
    { text: 'Help' }
  ];
  new Menu({ items: menuItems, cssClass: 'e-rounded-menu', showItemOnClick:
true }, '#menu');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/bootstrap.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/bootstrap.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/bootstrap.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <ul id="menu"></ul>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Ej1 api migration in EJ2 JavaScript Menu control

This article describes the API migration process of Menu component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Animation type on hover or click on the menu items | **Property:** *animationType*

 \$(("#menu").ejMenu({
animationType: ej.AnimationType.Default
 })); | Not applicable |

| Context menu target | **Property:** *contextMenuTarget*

 \$(("#menu").ejMenu({
menuType: ej.MenuType.ContextMenu,
contextMenuTarget: "#target",
 })); | Not applicable |

| Container element for submenu's collision | **Property:** *container*

 \$(("#menu").ejMenu({
container: "#target",
target: "#target"
 })); | Not applicable |

| Menu items | Not applicable | **Property:** *items*
 var menu = new ej.navigations.Menu({
items: menuItems
});
menu.appendTo("#menu");
 var menuItems = [

 text: 'File'
 },
 {
 text: 'Edit'
 },
 {
 text: 'More'
 },
 {
 text: 'Help'
 }
] |

| Adding custom class | **Property:** *cssClass*

 \$("#menu").ejMenu({
cssClass: "e-custom-class"
 }); | **Property:** *cssClass*

 var menu = new ej.navigations.Menu({
cssClass: "e-custom-class",
 items: menuItems
});
menu.appendTo("#menu"); |

| Enables or disables the animation on hover or click on the menu items | **Property:** *enableAnimation*

 \$("#menu").ejMenu({
enableAnimation: true,
 }); | Not applicable |

| Root menu items to be aligned center in horizontal menu | **Property:** *enableCenterAlign*

 \$("#menu").ejMenu({
enableCenterAlign: true,
 }); | Not applicable |

| Disabled state | **Property:** *enabled*

 \$("#menu").ejMenu({
enabled: false
 }); | **Property:** *disabled*

 var menu = new ej.navigations.Menu({
disabled: true,
 items: menuItems
});
menu.appendTo("#menu"); |

| RTL | **Property:** *enableRTL*

 \$("#menu").ejMenu({
enableRTL: true
 }); | **Property:** *enableRtl*

 var menu = new ej.navigations.Menu({
enableRtl: true,
 items: menuItems
});
menu.appendTo("#menu"); |

| Enables/Disables the separator | **Property:** *enableSeparator*

 \$("#menu").ejMenu({
enableSeparator: true
 }); | Not applicable |

| Exclude target for context menu | **Property:** *excludeTarget*

 \$("#menu").ejMenu({
excludeTarget: ".exclude-target",
menuType: ej.MenuType.ContextMenu,
contextMenuTarget: "#target"
 }); | Not applicable |

| Fields | **Property:** *fields*

 \$("#menu").ejMenu({
fields: menuFields
 }); | **Property:** *fields*

 var menu = new ej.navigations.Menu({
fields: menuFields,
 items: menuItems
});
menu.appendTo("#menu"); |

| Height | **Property:** *height*

 \$("#menu").ejMenu({
height: "25"
 }); | Not applicable |

| HTML Attributes | **Property:** *htmlAttributes*

 \$("#menu").ejMenu({
htmlAttributes: {"aria-label":"menu"}
 }); | Not applicable |

| Responsive | **Property:** *isResponsive*

 \$("#menu").ejMenu({
isResponsive: true
 }); | Not applicable |

| Menu Type | **Property:** *menuType*

 \$("#menu").ejMenu({
menuType: ej.MenuType.ContextMenu,
contextMenuTarget: "#target"
 }); | Not applicable |

| Show item on click | **Property:** *openOnClick*

 \$("#menu").ejMenu({
openOnClick: true
 }); | **Property:** *showItemOnClick*

 var menu = new ej.navigations.Menu({
showItemOnClick: true,
 items: menuItems
});
menu.appendTo("#menu"); |

| Orientation | **Property:** *orientation*

 \$("#menu").ejMenu({
orientation: ej.Orientation.Vertical
 }); | **Property:** *orientation*

 var menu = new ej.navigations.Menu({
orientation: "Vertical",
 items: menuItems
});
menu.appendTo("#menu"); |

| Show root level arrows | **Property:** *showRootLevelArrows*

 \$("#menu").ejMenu({
showRootLevelArrows: false
 }); | Not applicable |

| Show sub level arrows | **Property:** *showSubLevelArrows*

 \$("#menu").ejMenu({
showSubLevelArrows: false
 }); | Not applicable |

| Sub menu direction | **Property:** *subMenuDirection*

 \$("#menu").ejMenu({

subMenuDirection: "left"
 }); | Not applicable |

| Title | **Property:** *titleText*

 \$("#menu").ejMenu({
titleText: "Menu"
 }); | Not applicable |

| Width | **Property:** *width*

 \$("#menu").ejMenu({
width: "800px"
 }); | Not applicable |

| Animation settings | Not applicable | **Property:** *animationSettings*

 let animationSettings: MenuAnimationSettings = { effect: "FadeIn" }
 var menu = new ej.navigations.Menu({

animationSettings: animationSettings,
 items: menuItems
});

menu.appendTo("#menu"); |

| Template | Not applicable | **Property:** *template*

 var menu = new ej.navigations.Menu({
items: [{
 category: 'Services', options: [{
 suport: [
 { value: 'Application Development', count: '1200+' },
{ value: 'Maintenance & Support', count: '3700+' },
{ value: 'Quality Assurance' }
]
}
}
{ category: 'Careers' },
{ category: 'Sign In' }
],
template: "#template"}];
menu.appendTo("#menu"); |

| Pop up menu height | **Property:** *overflowHeight*

 \$("#menu").ejMenu({

overflowHeight: "200px"
 }); | Not applicable |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Disable Method | **Method:** *disable*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.disable(); | Not applicable |

| Disable menu items | **Method:** *disableItem*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.disableItem("Home"); | **Method:** *enableItems*

 var menu = new ej.navigations.Menu({
 items: menuItems
});

menu.appendTo("#menu");
 menu.enableItems("Home", false) |

| Disable menu items by ID | **Method:** *disableItemByID*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.disableItemByID("homeid"); | **Method:** *enableItems*

 var menu = new ej.navigations.Menu({
 items: menuItems
});

menu.appendTo("#menu");
 menu.enableItems("homeid", false, true) |

| Enable Method | **Method:** *enable*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.enable(); | Not applicable |

| Enable menu items | **Method:** *enableItem*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.enableItem("Home"); | **Method:** *enableItems*

 var menu = new ej.navigations.Menu({
 items: menuItems
});

menu.appendTo("#menu");
 menu.enableItems("Home", true); |

| Enable menu items by ID | **Method:** *enableItemByID*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.enableItemByID("homeid"); | **Method:** *enableItems*

 var menu = new ej.navigations.Menu({
 items: menuItems
});

menu.appendTo("#menu");
 menu.enableItems("homeid", true, true); |

| Hide Method | **Method:** *hide*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.hide(); | Not applicable |

| Hide menu items | **Method:** *hideItems*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.hideItems(["#search", "#company"]); | **Method:** *hideItems*

 var menu = new ej.navigations.Menu({
 items: menuItems
});
menu.appendTo("#menu");
 menu.hideItems(["search", "company"], true); |

| Insert menu items | **Method:** *insert*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.insert({ id: "More", text: "More" }, "#Home"); | Not applicable |

| Insert menu items after the specified menu item | **Method:** *insertAfter*

 \$("#menu").ejMenu({ });
 menu.insertAfter({ id: "More", text: "More" }, "#Home"); | **Method:** *insertAfter*

 var menu = new ej.navigations.Menu({
 items: menuItems
});
menu.appendTo("#menu");
 menu.insertAfter({ id: "More", text: "More" }, "Home"); |

| Insert menu items before the specified menu item | **Method:** *insertBefore*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.insertBefore({ id: "More", text: "More" }, "#Home"); | **Method:** *insertBefore*

 var menu = new ej.navigations.Menu({
 items: menuItems
});
menu.appendTo("#menu");
 menu.insertBefore({ id: "More", text: "More" }, "Home"); |

| Remove menu items | **Method:** *remove*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.remove(["#Home"]); | **Method:** *removeItems*

 var menu = new ej.navigations.Menu({
 items: menuItems
});
 menu.appendTo("#menu");
 menu.removeItems(["Home"]); |

| To show menu | **Method:** *show*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.show(); | Not applicable |

| To show menu items | **Method:** *showItems*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.showItems(["#search", "#company"]); | **Method:** *showItems*

 var menu = new ej.navigations.Menu({
 items: menuItems
});
menu.appendTo("#menu");
 menu.showItems(["Search", "Company"]); |

| Destroy method | **Method:** *destroy*

 \$("#menu").ejMenu({ });
 var menu = \$("#menu").data("ejMenu");
 menu.destroy(); | **Method:** *destroy*

 var menu = new ej.navigations.Menu({
 items: menuItems
});
 menu.appendTo("#menu");
 menu.destroy(); |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Triggers before opening the menu | **Events:** *beforeOpen*

 \$("#menu").ejMenu({
 beforeOpen: function(args) { / code block / }
 }); | **Events:** *beforeOpen*

 var menu = new ej.navigations.Menu({
 items: menuItems
 beforeOpen: function(args) { / code block / }
 });
menu.appendTo("#menu"); |

| Triggers before closing the menu | Not applicable | **Events:** *beforeClose*

 var menu = new ej.navigations.Menu({
 items: menuItems
 beforeClose: function(args) { / code block */ }
 });
menu.appendTo("#menu"); |

| Triggers before rendering each menu item | Not applicable | **Events:** *beforeItemRender*

var menu = new ej.navigations.Menu({
 items: menuitems
 beforeItemRender: function(args)
{ / code block */ }
});
menu.appendTo("#menu"); |

| Triggers while selecting the menu item | **Events:** *click*

 \$("#menu").ejMenu({
 click:
function(args) { / code block / }
 }); | **Events:** *select*

 var menu = new
**ej.navigations.Menu({
 items: menuitems
 select: function(args) { / code block / }
});**

menu.appendTo("#menu"); |

| Triggers after closing the menu | **Events:** *close*

 \$("#menu").ejMenu({
 close:
function(args) { / code block / }
 }); | **Events:** *onClose*

 var menu = new
**ej.navigations.Menu({
 items: menuitems
 onClose: function(args) { / code block / }
});**

menu.appendTo("#menu"); |

| Triggers after opening the menu | **Events:** *open*

 \$("#menu").ejMenu({
 open:
function(args) { / code block / }
 }); | **Events:** *onOpen*

 var menu = new
**ej.navigations.Menu({
 items: menuitems
 onOpen: function(args) { / code block / }
});**

menu.appendTo("#menu"); |

| Triggers once the component rendering is completed | **Events:** *create*

\$("#menu").ejMenu({
 create: function(args) { / code block / }
 }); | **Events:** *created*

 **var menu = new ej.navigations.Menu({
 items: menuitems
 created: function() {**
**/ code block / }
});**
menu.appendTo("#menu"); |

| Triggers once the component is destroyed | **Events:** *destroy*

 \$("#menu").ejMenu({

destroy: function(args) { / code block */ }
 }); | Not applicable |

| Triggers when key down on menu items | **Events:** *keydown*

 \$("#menu").ejMenu({

keydown: function(args) { / code block */ }
 }); | Not applicable |

| Triggers when mouse out from menu items | **Events:** *mouseout*

 \$("#menu").ejMenu({

mouseout: function(args) { / code block */ }
 }); | Not applicable |

| Triggers when mouse over the Menu items | **Events:** *mouseover*

\$("#menu").ejMenu({
 mouseover: function(args) { / code block */ }
 }); | Not applicable |

| Triggers when overflow popup menu opens | **Events:** *overflowOpen*

\$("#menu").ejMenu({
 overflowOpen: function(args) { / code block */ }
 }); | Not applicable |

| Triggers when overflow popup menu closes | **Events:** *overflowClose*

\$("#menu").ejMenu({
 overflowClose: function(args) { / code block */ }
 }); | Not applicable |

Message

Severities in EJ2 JavaScript Message control

The severity denotes the importance and context of the message to the user. The message contains different severity types. Use the [severity](#) property to display the messages with different severity levels.

The available severity types are **Normal**, **Success**, **Info**, **Warning** and **Error**. The default severity type for messages is **Normal**.

The following example demonstrates the severity of the messages.

INDEX.TS

```
import { Message } from '@syncfusion/ej2-notifications'
```

```
let msgDefault: Message = new Message({
    content: "Editing is restricted"
});
msgDefault.appendTo('#msg_default');
let msgInfo: Message = new Message({
    content: "Please read the comments carefully",
    severity: "Info"
});
msgInfo.appendTo('#msg_info');
let msgSuccess: Message = new Message({
    content: "Your message has been sent successfully",
    severity: "Success"
});
msgSuccess.appendTo('#msg_success');
let msgWarning: Message = new Message({
    content: "There was a problem with your network connection",
    severity: "Warning"
});
msgWarning.appendTo('#msg_warning');
let msgError: Message = new Message({
    content: "A problem occurred while submitting your data",
    severity: "Error"
});
msgError.appendTo('#msg_error');
```

INDEX.HTML

```
<html><head>
    <title>Essential JS 2 Message control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Message control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="msg-default-section">
            <div class="content-section">
                <div id="msg_default"></div>
                <div id="msg_info"></div>
                <div id="msg_success"></div>
                <div id="msg_warning"></div>
                <div id="msg_error"></div>
            </div>
        </div>
    </div>
```

```

<style>
  /* Sample level styles */
  .msg-default-section .content-section {
    margin: 0 auto;
    max-width: 450px;
    padding-top: 10px;
  }

  .msg-default-section .e-message {
    margin: 10px 0;
  }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Variants in EJ2 JavaScript Message control

The Message has predefined appearance variants for different visual representations. The variants of the message can be changed based on the [variant](#) property.

The available variants are **Text**, **Outlined** and **Filled**. The default variant type for messages is **Text**.

- **Text** - The severity is differentiated using a text color and a light background color.
- **Outlined** - The severity is differentiated using a text color and a border without a background.
- **Filled** - The severity is differentiated using a text color and a dark background color.

The following example demonstrates the default message with different variant types.

INDEX.TS

```

import { Message } from '@syncfusion/ej2-notifications'
let msgDefault: Message = new Message({});
msgDefault.appendTo('#msg_default');
let msgInfo: Message = new Message({
  severity: "Info"
});
msgInfo.appendTo('#msg_info');
let msgSuccess: Message = new Message({
  severity: "Success"
});
msgSuccess.appendTo('#msg_success');
let msgWarning: Message = new Message({
  severity: "Warning"
});
msgWarning.appendTo('#msg_warning');
let msgError: Message = new Message({
  severity: "Error"
});
msgError.appendTo('#msg_error');
let msgDefaultOutlined: Message = new Message({

```



```

        variant: "Outlined"
    });
    msgDefaultOutlined.appendTo('#msg_default_outlined');
    let msgInfoOutlined: Message = new Message({
        severity: "Info",
        variant: "Outlined"
    });
    msgInfoOutlined.appendTo('#msg_info_outlined');
    let msgSuccessOutlined: Message = new Message({
        severity: "Success",
        variant: "Outlined"
    });
    msgSuccessOutlined.appendTo('#msg_success_outlined');
    let msgWarningOutlined: Message = new Message({
        severity: "Warning",
        variant: "Outlined"
    });
    msgWarningOutlined.appendTo('#msg_warning_outlined');
    let msgErrorOutlined: Message = new Message({
        severity: "Error",
        variant: "Outlined"
    });
    msgErrorOutlined.appendTo('#msg_error_outlined');
    let msgDefaultFilled: Message = new Message({
        variant: "Filled"
    });
    msgDefaultFilled.appendTo('#msg_default_filled');
    let msgInfoFilled: Message = new Message({
        severity: "Info",
        variant: "Filled"
    });
    msgInfoFilled.appendTo('#msg_info_filled');
    let msgSuccessFilled: Message = new Message({
        severity: "Success",
        variant: "Filled"
    });
    msgSuccessFilled.appendTo('#msg_success_filled');
    let msgWarningFilled: Message = new Message({
        severity: "Warning",
        variant: "Filled"
    });
    msgWarningFilled.appendTo('#msg_warning_filled');
    let msgErrorFilled: Message = new Message({
        severity: "Error",
        variant: "Filled"
    });
    msgErrorFilled.appendTo('#msg_error_filled');

```

INDEX.HTML

```

<html><head>
  <title>Essential JS 2 Message control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Message control">
  <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="msg-variant-section">
            <div class="content-section">
                <h4>Filled</h4>
                <div id="msg_default_filled">Editing is restricted</div>
                <div id="msg_info_filled">Please read the comments
carefully</div>
                <div id="msg_success_filled">Your message has been sent
successfully</div>
                <div id="msg_warning_filled">There was a problem with your
network connection</div>
                <div id="msg_error_filled">A problem occurred while
submitting your data</div>
            </div>
            <div class="content-section">
                <h4>Outlined</h4>
                <div id="msg_default_outlined">Editing is restricted</div>
                <div id="msg_info_outlined">Please read the comments
carefully</div>
                <div id="msg_success_outlined">Your message has been sent
successfully</div>
                <div id="msg_warning_outlined">There was a problem with your
network connection</div>
                <div id="msg_error_outlined">A problem occurred while
submitting your data</div>
            </div>
            <div class="content-section">
                <h4>Text</h4>
                <div id="msg_default">Editing is restricted</div>
                <div id="msg_info">Please read the comments carefully</div>
                <div id="msg_success">Your message has been sent
successfully</div>
                <div id="msg_warning">There was a problem with your network
connection</div>
                <div id="msg_error">A problem occurred while submitting your
data</div>
            </div>
        </div>
    </div>
</div>
<style>
    /* Sample level styles */
    .msg-variant-section .content-section {
        margin: 0 auto;
        max-width: 520px;
        padding: 10px;
    }

```

```

    }
    .msg-variant-section .e-message {
        margin: 10px 0;
    }
    .msg-variant-section {
        display: flex;
    }
}
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Icons in EJ2 JavaScript Message control

This section explains the message with no icons, how to show or hide the close icon and add the custom severity icon to the message.

No Icon

By default, severity icons can be displayed according to the severity types to make it more understandable to the user by visual information rather than text. To hide the severity icons, set the [showIcon](#) property to `false`.

The following example demonstrates the different severity messages without the severity icons.

INDEX.TS

```

import { Message } from '@syncfusion/ej2-notifications'
let msgDefault: Message = new Message({
    content: "Editing is restricted",
    showIcon: false
});
msgDefault.appendTo('#msg_default');
let msgInfo: Message = new Message({
    content: "Please read the comments carefully",
    severity: "Info",
    showIcon: false
});
msgInfo.appendTo('#msg_info');
let msgSuccess: Message = new Message({
    content: "Your message has been sent successfully",
    severity: "Success",
    showIcon: false
});
msgSuccess.appendTo('#msg_success');
let msgWarning: Message = new Message({
    content: "There was a problem with your network connection",
    severity: "Warning",
    showIcon: false
});
msgWarning.appendTo('#msg_warning');
let msgError: Message = new Message({
    content: "A problem occurred while submitting your data",

```

```

        severity: "Error",
        showIcon: false
    });
    msgError.appendTo('#msg_error');

```

INDEX.HTML

```

<html><head>
    <title>Essential JS 2 Message control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Message control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="msg-default-section">
            <div class="content-section">
                <div id="msg_default"></div>
                <div id="msg_info"></div>
                <div id="msg_success"></div>
                <div id="msg_warning"></div>
                <div id="msg_error"></div>
            </div>
        </div>
    </div>
</div>
<style>
    /* Sample level styles */
    .msg-default-section .content-section {
        margin: 0 auto;
        max-width: 450px;
        padding-top: 10px;
    }

    .msg-default-section .e-message {
        margin: 10px 0;
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom Icon

By default, the severity icons can be displayed according to the severity type to make it more understandable to the user by visual information rather than text. If the user wants to customize these icons, it can be achieved through the `cssClass` property.

The following example demonstrates how the default message is rendered with a custom severity icon.

INDEX.TS

```
import { Message } from '@syncfusion/ej2-notifications'
let msgIcon: Message = new Message({
  cssClass: "custom"
});
msgIcon.appendTo('#msg_icon');
```

INDEX.HTML

```
<html><head>
  <title>Essential JS 2 Message control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Message control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="msg-custom-section">
      <div class="content-section">
        <div id="msg_icon">Essential JS 2 is a modern JavaScript UI
Controls library built from the ground up to be lightweight, responsive,
modular, and touch friendly. It is written in the TypeScript and has no
external dependencies. It also includes complete support for Angular, React,
Vue, ASP.NET MVC, and ASP.NET Core frameworks.</div>
      </div>
    </div>
  </div>
  <style>
    /* Sample level styles */
    .msg-custom-section .content-section {
      margin: 0 auto;
      max-width: 400px;
      padding-top: 10px;
    }
    .msg-custom-section .e-message {
```

```

        margin: 10px 0;
    }
    @font-face {
        font-family: 'Message_icons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj8BS4YAAAEoAAAAVmNtYXDopOjpAAABiAAAADZnbHlmEsN
XoQAAAcgAAAjMAgVhZCGnOH4AAADQAAAAANmhoZWEIPwQDAAAARAAAACRobXR4CAAAAAAAYAAAAA
IbG9jYQEmAAAAAHAAAAABm1heHABEAEoAAABCAAAACBuYw1l1dofAwAABBQAAAJtcG9zdNGGZXAA
AAAAeAAAAALwABAAAEAAAAAFwEAAAAAAD4gABAAAAAAGABAAAAAQAAecv2N18
PPPUACwQAAAAAAN9TeeEAAAAA31N54QAAAAAD4gO/AAAACAACAAAAAEEAAAAACAQIABAAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAQAAZABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA6JTolAQAAAAAXAQAAAAAABAAAAAABAAAAAQAQAAA
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA6JT//wAA6JT//wAAAAEABAAAAEAAAA
AAAABJgAAAAQAAAAA+IDvwANAG4AoAEBAABLwMrATUzPwQXDwMVHxIdAQ8SFR8GMz8RPQEVEsM
PAiCHIw8HERUFDD8JES8JIw8BNw8DFR8SHQEPEhUfBj8SPQEVEsMPAgHCjwYHCAelpQcIBwAPqQM
CAwECAwUGCQkIBwCHBQYEBQMDAgIBAQICAwMFBAYFBwCHCAkJBgUDAgEDBQcICQkHBGcGDQwLCwo
JCAChBgUEAwICAgIDBAUGBwCICQoLCwwNBgcHBwkJB6LJvwkIBwYFBAIBAwQFBgcECMTJBgcHCAg
JBQUEBAMCAQEBAQIDBAQFBQkJBwCH/QMCAwECAwUGFRQSEREODgwLCggGBgMDAwMGBggKCwwODhE
REhQVBgUDAgEDBQcICQkHBGcGGBgWFBMSDw8NCwkIBgUDAwUGCAkLDQ8PEhMUFhgYBgcHBwkJBwE
lfwQEAggBAGMffwQDBAgICAgHBwYHCAkJCQkKCgsLCgwLDASMDAsMCwwKCwsKCgkJCQkIBwYHBwg
ICAgHBwUDAQIDBAoMDAwNDQ4PDg8QEBAQEBEREBAQEBAQPDg8ODQ0MDAwKBAMCAQMFErMBAGQFBgc
ECP7/CAgHBgYDAGEBsgUDAgEBAwMEBAUFbQYGANyGBgYFBQQAeAwIBAgROBAMICAgIBwCGERMTFRU
WFxcYGRKaGhsbGxsbGxoaGRkYFxcWFRUTEExEGBwCICAgIBwCFAwEBAQMEFRUXGBkaGxsdHR4eHx8
gICAgHx8eHh0dGxsaGRgXFRUEAwIBAwUAAAAAEgDeAAEAAAAAAAAAAAAQAAAAEAAAAAAAAEADQABAAE
AAAAAAAAIABwAOAAEAAAAAAAAAMADQAVAAEAAAAAAAAAQADQAIAAEAAAAAAAAUACwAvAAEAAAAAAAAAYADQA
6AAEAAAAAAAAoALABHAAEAAAAAAAAsAEgBzAAMAAQQAQAAAAgCFAAMAAQQAQAAEAGgCHAAMAAQQAQAAI
ADgChAAMAAQQAAMAGgCvAAMAAQQAQAAQAGgDJAAMAAQQAQAAUAFgDJAAMAAQQAQAAAYAGgD5AAMAAQQA
JAAoAWAETAAMAAQQAQAAAJAFrIE1lc3NhZ2VfaWNvbnNSZWdlbGZyTWVzc2FnZV9pY29uc01lc3N
hZ2VfaWNvbnNWZXJzaW9uIDEuME1lc3NhZ2VfaWNvbnNGb250IGdlbmVyYXRlZCB1c2luZyBTZW5j
ZnZzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnZzaW9uLmNvbQAgAE0AZQBzAHMAYQBnAGUAXwB
pAGMABwBuAHMAUgBlAGCAAdQBsAGEAcgBNAGUAcwBzAGEAZwBlAF8AaQBjAG8AbgBzAE0AZQBzAHM
AYQBnAGUAXwBpAGMABwBuAHMAVgBlAHIAcWBPAG8AbgAgADEALgAwAE0AZQBzAHMAYQBnAGUAXwB
pAGMABwBuAHMARgBvAG4AdAAgAGcAZQBwAGUAcgBhAHQAZQBkACAAdQBsAGkAbgBnACAAUwB5AG4
AYwBmAHUAcWBPAG8AbgAgAE0AZQB0AHIAbWAgAFMAdAB1AGQAAQBVAHcAdwB3AC4AcwB5AG4AYwB
mAHUAcWBPAG8AbgAuAGMABwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgE
CAQMABWF1ZGlVAAAA) format('trueType');
        font-weight: normal;
        font-style: normal;
    }
    .custom.e-message .e-msg-icon::before {
        font-family: 'Message_icons';
        content: '\e894';
    }
</style>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Close Icon

The message can be rendered with or without the close icon. The close icon is used to hide the message, either by manually clicking the close icon or through keyboard interaction.

By default, the close icon is not rendered in the message. To show the close icon, set the [showCloseIcon](#) property to **true**.

In the following example, the messages are rendered with the close icon.

INDEX.TS

```
import { getComponent } from '@syncfusion/ej2-base';
import { CheckBox, ChangeEventArgs, Button } from '@syncfusion/ej2-buttons';
import { Message } from '@syncfusion/ej2-notifications';
let showButton: Button = new Button({ content: 'Show default message',
cssClass: "e-outline e-primary msg-hidden" });
showButton.appendTo('#btn1');
showButton.element.onclick = (): void => {
    show(msgDefaultIcon, showButton);
}
let infoButton: Button = new Button({ content: 'Show info message',
cssClass: "e-outline e-primary e-info msg-hidden" });
infoButton.appendTo('#btn2');
infoButton.element.onclick = (): void => {
    show(msgInfoIcon, infoButton);
}
let successButton: Button = new Button({ content: 'Show success
message', cssClass: "e-outline e-primary e-success msg-hidden" });
successButton.appendTo('#btn3');
successButton.element.onclick = (): void => {
    show(msgSuccessIcon, successButton);
}
let warningButton: Button = new Button({ content: 'Show warning
message', cssClass: "e-outline e-primary e-warning msg-hidden" });
warningButton.appendTo('#btn4');
warningButton.element.onclick = (): void => {
    show(msgWarningIcon, warningButton);
}
let errorButton: Button = new Button({ content: 'Show error message',
cssClass: "e-outline e-primary e-danger msg-hidden" });
errorButton.appendTo('#btn5');
errorButton.element.onclick = (): void => {
    show(msgErrorIcon, errorButton);
}
let msgDefaultIcon: Message = new Message({
    content: "Editing is restricted",
    showCloseIcon: true,
    closed: () => {
        showButton.element.classList.remove('msg-hidden');
    }
});
msgDefaultIcon.appendTo('#msg_default_icon');
let msgInfoIcon: Message = new Message({
    content: "Please read the comments carefully",
    severity: "Info",
    showCloseIcon: true,
```

```

        closed: () => {
            infoButton.element.classList.remove('msg-hidden');
        }
    });
    msgInfoIcon.appendTo('#msg_info_icon');
    let msgSuccessIcon: Message = new Message({
        content: "Your message has been sent successfully",
        severity: "Success",
        showCloseIcon: true,
        closed: () => {
            successButton.element.classList.remove('msg-hidden');
        }
    });
    msgSuccessIcon.appendTo('#msg_success_icon');
    let msgWarningIcon: Message = new Message({
        content: "There was a problem with your network connection",
        severity: "Warning",
        showCloseIcon: true,
        closed: () => {
            warningButton.element.classList.remove('msg-hidden');
        }
    });
    msgWarningIcon.appendTo('#msg_warning_icon');
    let msgErrorIcon: Message = new Message({
        content: "A problem occurred while submitting your data",
        severity: "Error",
        showCloseIcon: true,
        closed: () => {
            errorButton.element.classList.remove('msg-hidden');
        }
    });
    msgErrorIcon.appendTo('#msg_error_icon');
    function show(message: Message, btn: Button): void {
        message.visible = true;
        btn.element.classList.add('msg-hidden');
    }
}

```

INDEX.HTML

```

<html><head>
    <title>Essential JS 2 Message control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Message control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

<div id="container">
  <div class="msg-icon-section">
    <div class="content-section">
      <button id="btn1"></button>
      <div id="msg_default_icon"></div>
      <button id="btn2"></button>
      <div id="msg_info_icon"></div>
      <button id="btn3"></button>
      <div id="msg_success_icon"></div>
      <button id="btn4"></button>
      <div id="msg_warning_icon"></div>
      <button id="btn5"></button>
      <div id="msg_error_icon"></div>
    </div>
  </div>
</div>
<style>
  /* Sample level styles */
  .msg-icon-section .content-section {
    margin: 0 auto;
    max-width: 450px;
    padding-top: 10px;
  }
  .msg-icon-section .e-message {
    margin: 10px 0;
  }
  .msg-icon-section .e-btn {
    display: block;
    margin: 10px 0;
  }
  .msg-icon-section .e-btn.msg-hidden {
    display: none;
  }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization in EJ2 JavaScript Message control

The Message control allows the user to customize the content display positions and appearance. This section explains the details about changing the content alignments and border styles for messages.

Content Alignment

Normally, the message content is aligned to the **left**. The Message control allows the user to align the message content in the **center** or **right** through the built-in classes `e-content-center` and `e-content-right`.

The following example demonstrates the message with different content alignments.

INDEX.TS

```
import { Message } from '@syncfusion/ej2-notifications'
let msgLeft: Message = new Message({
  content: "Your license has been activated successfully",
  severity: "Success"
});
msgLeft.appendTo('#msg_content_left');
let msgCenter: Message = new Message({
  content: "The license will expire today",
  cssClass: "e-content-center",
  severity: 'Warning'
});
msgCenter.appendTo('#msg_content_center');
let msgRight: Message = new Message({
  content: "The license key is invalid",
  cssClass: "e-content-right",
  severity: 'Error'
});
msgRight.appendTo('#msg_content_right');
```

INDEX.HTML

```
<html><head>
  <title>Essential JS 2 Message control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Message control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="msg-custom-section">
      <div class="content-section">
        <h4>Content Alignment</h4>
        <div id="msg_content_left"></div>
        <div id="msg_content_center"></div>
        <div id="msg_content_right"></div>
      </div>
    </div>
  </div>
  <style>
    /* Sample level styles */
    .msg-custom-section .content-section {
      margin: 0 auto;
      max-width: 400px;
```

```

        padding-top: 10px;
    }
    .msg-custom-section .e-message {
        margin: 10px 0;
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Rounded and Square

To customize the Message control's appearance, add the custom class to the message through the [cssClass](#) property. This custom class will be added to the root element. Based on this custom class, the user can override the message styles at the application level.

The following example shows the rounded and squared appearance of the message, which can be achieved by adding the `cssClass` property.

INDEX.TS

```

import { Message } from '@syncfusion/ej2-notifications'
let msgRounded: Message = new Message({
    content: "The license will expire today",
    severity: 'Warning',
    cssClass: "rounded"
});
msgRounded.appendTo('#msg_rounded');
let msgSquare: Message = new Message({
    content: "The license key is invalid",
    severity: 'Error',
    cssClass: "square"
});
msgSquare.appendTo('#msg_square');

```

INDEX.HTML

```

<html><head>
    <title>Essential JS 2 Message control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Message control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="msg-custom-section">
            <div class="content-section">
                <h4>Rounded</h4>
                <div id="msg_rounded"></div>
                <h4>Square</h4>
                <div id="msg_square"></div>
            </div>
        </div>
    </div>
</div>
<style>
    /* Sample level styles */
    .msg-custom-section .content-section {
        margin: 0 auto;
        max-width: 400px;
        padding-top: 10px;
    }
    .msg-custom-section .e-message {
        margin: 10px 0;
    }
    .msg-custom-section .e-message.rounded {
        border-radius: 5px;
    }
    .msg-custom-section .e-message.square {
        border-radius: 1px;
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

CSS Message

The Essential JS 2 Message has predefined CSS classes that can be defined in the HTML elements, which renders the message without any script reference. This can display a simple message with content and make the code lighter.

The following DOM structure is required to display the simple message with the content.

```
`bash
```

```

<div class="e-message">
<div class="e-msg-content">..content..</div>
</div>
`

```

The following DOM structure is required to display the simple message with the content and severity icon.

```
`bash
<div class="e-message">
<span class="e-msg-icon"></span>
<div class="e-msg-content">..content..</div>
</div>
`
```

The following is the available list of predefined CSS classes to make the appearance of a message.

Class	Description
-----	-----
e-message	Represents the message wrapper.
e-msg-icon	Represents the severity type icon.
e-msg-content	Represents the message content.
e-msg-close-icon	Represents the close icon.
e-info	Represents the information message.
e-success	Represents the success message.
e-warning	Represents the warning message.
e-error	Represents the error message.
e-content-center	Aligns the message content to the center.
e-content-right	Aligns the message content to the right.

The following example shows the message which renders without any script reference.

INDEX.HTML

```
<html><head>
  <title>Essential JS 2 Message control</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Message control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
```

```

        <div class="msg-default-section">
            <div class="content-section">
                <div id="msg-default" class="e-message" role="alert">
                    <span class="e-msg-icon"></span>
                    <div class="e-msg-content">Editing is restricted</div>
                </div>
                <div id="msg-info" class="e-message e-info" role="alert">
                    <span class="e-msg-icon"></span>
                    <div class="e-msg-content">Please read the comments
carefully</div>
                </div>
                <div id="msg-success" class="e-message e-success"
role="alert">
                    <span class="e-msg-icon"></span>
                    <div class="e-msg-content">Your message has been sent
successfully</div>
                </div>
                <div id="msg-warning" class="e-message e-warning"
role="alert">
                    <span class="e-msg-icon"></span>
                    <div class="e-msg-content">There was a problem with your
network connection</div>
                </div>
                <div id="msg-error" class="e-message e-error" role="alert">
                    <span class="e-msg-icon"></span>
                    <div class="e-msg-content">A problem occurred while
submitting your data</div>
                </div>
            </div>
        </div>
        <style>
            /* Sample level styles */
            .msg-default-section .content-section {
                margin: 0 auto;
                max-width: 450px;
                padding-top: 10px;
            }

            .msg-default-section .e-message {
                margin: 10px 0;
            }
        </style>
        <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
        </script>
        <script src="index.js" type="text/javascript"></script>
    </body></html>

```

Template in EJ2 JavaScript Message control

The message supports templates that allows the user to customize the content with a custom structure. The content can be a string, paragraph, or any other HTML element. The template can be rendered through the [content](#) property or added directly to the HTML element.

In the following sample, the Message control content is customized with HTML elements and JavaScript Button controls, which are directly added to the HTML element.

INDEX.TS

```
import { Message } from '@syncfusion/ej2-notifications';
import { Button } from '@syncfusion/ej2-buttons';
let showButton: Button = new Button({ content: 'Show pull request',
cssClass: "e-outline e-primary e-success msg-hidden" });
showButton.appendTo('#btn');
showButton.element.onclick = (): void => {
    msgTemplate.visible = true;
    showButton.element.classList.add('msg-hidden');
}
let msgTemplate: Message = new Message({
    severity: "Success",
    closed: () => {
        showButton.element.classList.remove('msg-hidden');
    }
});
msgTemplate.appendTo('#msg_template');
let button: Button = new Button({ cssClass: 'e-link', content: 'Dismiss'
});
button.appendTo('#closeBtn');
button.element.onclick = (): void => {
    msgTemplate.visible = false;
}
let commitButton: Button = new Button({ cssClass: 'e-link', content:
'View commit' });
commitButton.appendTo('#commitBtn');
```

INDEX.HTML

```
<html><head>
    <title>Essential JS 2 Message control</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Message control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
  <div class="msg-template-section">
    <div class="content-section">
      <button id="btn"></button>
      <div id="msg_template">
        <p class="title">Merged pull request</p>
        <p>Pull request #41 merged after a successful build</p>
        <button id="commitBtn"></button>
        <button id="closeBtn"></button>
      </div>
    </div>
  </div>
</div>
<style>
  /* Sample level styles */
  .msg-template-section .content-section {
    margin: 0 auto;
    max-width: 450px;
    padding-top: 20px;
  }
  .msg-template-section .e-btn.msg-hidden {
    display: none;
  }
  .msg-template-section .e-message .title {
    margin: 0;
    font-size: 16px;
    font-weight: 600;
    line-height: 1.25;
  }
  .msg-template-section .e-message .e-msg-icon {
    padding: 0 4px;
    margin-top: 3px;
  }
  .msg-template-section .e-message p {
    margin: 8px 0 4px;
  }
  .msg-template-section .e-message .e-btn {
    padding: 0;
  }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in EJ2 JavaScript Message control

The Message control followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Message control is outlined below.

| Accessibility Criteria | Compatibility |


```
| -- | -- |
| WCAG 2.2 Support |  |
| Section 508 Support |  |
| Screen Reader Support |  |
| Right-To-Left Support |  |
| Color Contrast |  |
| Mobile Device Support |  |
| Keyboard Navigation Support |  |
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the control meet the requirement.</div>
<div> - Some features of the control do not meet the requirement.</div>
<div> - The control does not meet the requirement.</div>
```

WAI-ARIA attributes

The Message control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Message control:

```
| Attributes | Purpose |
| --- | --- |
| role=alert | Used to convey a significant and contextual message to the user. |
```

| **aria-label** | Provides an accessible name for the close icon. |

Keyboard interaction

The Message control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Message control.

| **Press** | **To do this** |

| --- | --- |

| **Tab / Shift + Tab** | To focus the close icon in the message. |

| **Enter / Space** | Closes the focused close icon's message. |

Ensuring accessibility

The Message control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Message control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Message control with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript controls](#)

MultiSelect

Tags in EJ2 JavaScript Multi select control

The MultiSelect can be initialized on three different tags as described in below. Though it is initialized in different tags, the UI appearance and built-in features behave in the same way.

Select element

When a MultiSelect is initialized on SELECT element, the list items can be assigned through the option tag of the HTML select element.

- The nested items are wrapped and grouped based on the `group` tag that is available

within the `<select>` element, by default.

- You can preselect the option by setting the `selected` attribute to an option tag.

INDEX.TS

```
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
// initialize MultiSelect component
let msObject: MultiSelect = new MultiSelect({
  placeholder: "Select vegetables"
});
// render initialized MultiSelect
msObject.appendTo('#selectElement');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <select id="selectElement">
      <optgroup label="Beans">
        <option value="1">Chickpea</option>
        <option value="2">Green bean</option>
        <option value="3">Horse gram</option>
      </optgroup>
      <optgroup label="Leafy and Salad">
        <option value="4" selected="selected">Cabbage</option>
        <option value="5">Spinach</option>
        <option value="6">Wheat grass</option>
      </optgroup>
    </select>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

UL element

The MultiSelect can be initialized through `` element which contains a collection of `` element. The `` items act as a popup list items of the MultiSelect. The inner text of the `` element is considered both as text and value fields.

INDEX.TS

```
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
```

```
// initialize MultiSelect component
let msObject: MultiSelect = new MultiSelect({
  placeholder:"Select vegetables"
});
// render initialized MultiSelect
msObject.appendTo('#ulElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <ul id="ulElement">
      <li>Cabbage</li>
      <li>Spinach</li>
      <li>Wheat grass</li>
    </ul>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Input element

The MultiSelect has also be rendered through `<input>` element with an array of either simple or complex data that is set through the [dataSource](#) property. It can retrieve data from local data sources as well as remote data services.

Detailed information about the data binding with an example is available in: [Data Binding to MultiSelect](#)

Data binding in EJ2 JavaScript Multi select control

The MultiSelect loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of `array` or `DataManager`.

The MultiSelect also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON, and JSONP with the help of `DataManager` adaptors.

Fields	Type	Description
text	string	Specifies the display text of each list item.
value	number or string	Specifies the hidden data value mapped to each list item that should contain a unique value.
groupBy	string	Specifies the category under which the list item has to be grouped.
iconCss	string	Specifies the icon class of each list item.

When binding complex data to the MultiSelect, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in two ways as described below.

1. Array of string

The MultiSelect has support to load array of primitive data such as strings and numbers. Here, both value and text field act the same.

INDEX.TS

```
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
// define the array of data
let sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize MultiSelect component
let msObject: MultiSelect = new MultiSelect({
    //set the data to dataSource property
    dataSource: sportsData,
    // set placeholder to multiSelect input element
    placeholder: "Select games"
});
// render initialized multiSelect
msObject.appendTo('#select');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="select">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

2. Array of object

The MultiSelect can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, `id` column and `sports` column from complex data have been mapped to the `value` field and `text` field, respectively.

INDEX.TS

```

import { MultiSelect } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
    { id: 'game1', sports: 'Badminton' },
    { id: 'game2', sports: 'Football' },
    { id: 'game3', sports: 'Tennis' }
];
//initiate the MultiSelect
let msObject: MultiSelect = new MultiSelect({
    // bind the sports Data to datasource property
    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { text: 'sports', value: 'id' },
    //set the placeholder to MultiSelect input
    placeholder:"Select games"
});
//render the component
msObject.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <!--element which is going to render the MultiSelect-->
    <input type="text" tabindex="1" id="select">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

3. Array of complex object

The MultiSelect can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Code.Id** column and **Country.Name** column from complex data have been mapped to the **value** field and **text** field, respectively.

INDEX.TS

```

import { MultiSelect } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let countriesData: { [key: string]: Object }[] = [
  { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
  { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
  { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
  { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },

```

```

        { Country:{Name: 'Denmark'}, Code:{ Id: 'DK' }},
        { Country:{Name: 'France'}, Code: { Id:'FR' } }
    ];
    //initiate the MultiSelect
    let msObject: MultiSelect = new MultiSelect({
        // bind the sports Data to datasource property
        dataSource: countriesData,
        // maps the appropriate column to fields property
        fields: { text: 'Country.Name', value: 'Code.Id' },
        //set the placeholder to MultiSelect input
        placeholder:"Select a country"
    });
    //render the component
    msObject.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 MultiSelect</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <!--element which is going to render the MultiSelect-->
        <input type="text" tabindex="1" id="select">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


Binding remote data

The MultiSelect supports retrieval of data from remote data services with the help of DataManager component. The [Query](#) property is used to fetch data from the database and bind it to the MultiSelect.

The following sample displays the first 6 contacts from “Customers” table of the **Northwind** Data Service.

INDEX.TS

```
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let customers: MultiSelect = new MultiSelect({
    //bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'ContactName', value: 'CustomerID' },
    //set the placeholder to MultiSelect input
    placeholder: "Select customers",
    //sort the resulted items
    sortOrder: 'Ascending'
});
//render the component
customers.appendTo('#select');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <!--element which is going to render the MultiSelect-->
        <input type="text" tabindex="1" id="select">
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to load data using template](#)
- [How to group the data using header](#)
- [How to filter the bound data](#)

Templates in EJ2 JavaScript Multi select control

The MultiSelect has been provided with several options to customize each list item, group title, selected value, header, and footer elements. It uses the Essential JS 2 [Template engine](#) to compile and render the elements properly.

Item template

The content of each list item within the MultiSelect can be customized with the help of [itemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data's.

INDEX.TS

```

import { MultiSelect } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let msObject: MultiSelect = new MultiSelect({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },

```

```
//set the placeholder to MultiSelect input
placeholder:"Select an employee",
//sort the resulted items
sortOrder: 'Ascending',
//set the value to itemTemplate property
itemTemplate:"<div><span class='name'>${FirstName}</span><span class
='city'>${City}</span></div>"
});
//render the component
msObject.appendTo('#select');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="select">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Value template

The currently selected value that is displayed by default on the MultiSelect input element can be customized using the [valueTemplate](#) property.

In the following sample, the selected value is displayed as a combined text of both **FirstName** and **City** in the MultiSelect input, which is separated by a hyphen.

INDEX.TS

```
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let msObject: MultiSelect = new MultiSelect({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },
    //set the placeholder to MultiSelect input
    placeholder: "Select an employee",
    mode: 'Box',
    //sort the resulted items
    sortOrder: 'Ascending',
    //set the template value to itemTemplate property
    itemTemplate: "<div><span class='name'>${FirstName}</span><span class
='city'>${City}</span></div>",
    //set the value to valueTemplate property
    valueTemplate: "<span>${FirstName} - ${City}</span>"
});
//render the component
msObject.appendTo('#select');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 MultiSelect</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```

<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="select">
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Group template

The group header title under which appropriate sub-items are categorized can also be customize with the help of [groupTemplate](#) property. This template is common for both inline and floating group header template.

In the following sample, employees are grouped according to their city.

INDEX.TS

```

import { MultiSelect } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, Predicate , DataManager, ODataV4Adaptor } from
 '@syncfusion/ej2-data';
// form predicate to fetch the grouped data
let groupPredicate = new Predicate('City',
'equal','london').or('City','equal','seattle');
//initiates the component
let msObject: MultiSelect = new MultiSelect({
  //bind the data manager instance to dataSource property
  dataSource: new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(5)
  .where(groupPredicate),
  //map the appropriate columns to fields property
  fields: { text: 'FirstName', value: 'EmployeeID', groupBy: 'City' },
  //set the placeholder to MultiSelect input
  placeholder: "Select employees",
  //sort the resulted items
  sortOrder: 'Ascending',
  //set the value to groupTemplate
  groupTemplate: "<strong>${City}</strong>"
});
//render the component
msObject.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="select">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Header template

The header element is shown statically at the top of the popup list items within the MultiSelect, and any custom element can be placed as a header element using the [headerTemplate](#) property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

INDEX.TS

```

import { MultiSelect } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let msObject: MultiSelect = new MultiSelect({
  //bind the data manager instance to dataSource property
  dataSource: new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  }),

```

```

    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },
    //set the placeholder to MultiSelect input
    placeholder:"Select employees",
    //sort the resulted items
    sortOrder: 'Ascending',
    //set the value to header template
    headerTemplate:"<span class='head'><span class='name'>Name</span><span
class='city'>City</span></span>",
    //set the value to item template
    itemTemplate:"<span class='item' ><span
class='name'>${FirstName}</span><span class='city'>${City}</span></span>"
});
//render the component
msObject.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <input type="text" id="select">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Footer template

The MultiSelect has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the [footerTemplate](#) property.

In the following sample, footer element displays the total number of list items present in the MultiSelect.

INDEX.TS

```
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
let sportsData = [ "Basketball", "Cricket", "Football", "Golf"];
//initiates the component
let msObject: MultiSelect = new MultiSelect({
    //bind the data manager instance to dataSource property
    dataSource: sportsData ,
    //set the placeholder to MultiSelect input
    placeholder:"Select games",
    //set the value to footer template
    footerTemplate:"<span class='foot'> Total list items: "+
sportsData.length + "</span>"
});
//render the component
msObject.appendTo('#select');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">

    <input type="text" id="select">
  </div>
</body>
</html>
```



```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

No records template

The MultiSelect is provided with support to custom design the popup list content when no data is found and no matches found on search with the help of [noRecordsTemplate](#) property.

In the following sample, popup list content displays the notification of no data available.

INDEX.TS

```

import { MultiSelect } from '@syncfusion/ej2-dropdowns';
//initiates the component
let msObject: MultiSelect = new MultiSelect({
    //bind the data manager instance to dataSource property
    dataSource: [],
    //set the placeholder to MultiSelect input
    placeholder: "Select an item",
    //set the value to noRecords template
    noRecordsTemplate: "<span class='norecord'> NO DATA AVAILABLE</span>"
});
//render the component
msObject.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 MultiSelect</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">

```

```

        <input type="text" tabindex="1" id="select">
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Action failure template

There is also an option to custom design the popup list content when the data fetch request fails at the remote server. This can be achieved using the [actionFailureTemplate](#) property.

In the following sample, when the data fetch request fails, the MultiSelect displays the notification.

INDEX.TS

```

import { MultiSelect } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let MultiSelectObject: MultiSelect = new MultiSelect({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        // Here, use the wrong url to display the action failure
        template
        url: 'https://services.odata.org/V4/Northwind/Northwind.svcs/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },
    //set the placeholder to MultiSelect input
    placeholder: "Select an employee",
    //set the value to action failure template
    actionFailureTemplate: "<span class='action-failure'> Data fetch get
fails</span>"
});
//render the component
MultiSelectObject.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 MultiSelect</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="select">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to bind the data](#)
- [How to group the data using header](#)
- [How to customize the options in MultiSelect](#)

Grouping in EJ2 JavaScript Multi select control

The MultiSelect supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the [groupBy](#) field in the data table. The group header is displayed both as inline and fixed headers. The fixed group header content is updated dynamically on scrolling the popup list with its category value.

In the following sample, vegetables are grouped according on its category using `groupBy` field.

INDEX.TS

```

import { MultiSelect } from '@syncfusion/ej2-dropdowns';
//define the data with category
let vegetableData: { [key: string]: Object }[] = [
    { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
    { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },
    { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3' },
},
    { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
    { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },
    { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
    { vegetable: 'Green bean', category: 'Beans', id: 'item7' },

```

```

        { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
        { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
        { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
        { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
    ];
    //initiate the MultiSelect
    let vegetables: MultiSelect = new MultiSelect({
        //set the grouped data to dataSource property
        dataSource: vegetableData,
        // map the groupBy field with category column
        fields: { groupBy: 'category', text: 'vegetable', value: 'id' },
        // set the placeholder to the MultiSelect input
        placeholder: "Select vegetables",
        // Set the popup list height
        popupHeight: '200px'
    });
    //render the MultiSelect component
    vegetables.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 MultiSelect</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <!--element which is going to render the MultiSelect-->
        <input type="text" tabindex="1" id="select">
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

HTML select

The MultiSelect also supports grouping of list items under specific groups by initiating the `<select>` element using `optgroup`. The nested items are wrapped based on the `<optgroup>` tag that is presents in the `<select>` element

```
<select id="selectElement">
  <optgroup label="Beans">
    <option value="1">Chickpea</option>
    <option value="2">Green bean</option>
    <option value="3">Horse gram</option>
  </optgroup>
  <optgroup label="Leafy and Salad">
    <option value="4">Spinach</option>
    <option value="5" selected="selected">Cabbage</option>
    <option value="6">Wheat grass</option>
  </optgroup>
</select>
```

INDEX.TS

```
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
// initialize MultiSelect component
let msObject: MultiSelect = new MultiSelect({
  placeholder:"Select vegetables"
});
// render initialized MultiSelect
msObject.appendTo('#selectElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <select id="selectElement">
            <optgroup label="Beans">
                <option value="1">Chickpea</option>
                <option value="2">Green bean</option>
                <option value="3">Horse gram</option>
            </optgroup>
            <optgroup label="Leafy and Salad">
                <option value="4" selected="selected">Cabbage</option>
                <option value="5">Spinach</option>
                <option value="6">Wheat grass</option>
            </optgroup>

        </select>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

The grouping header is also provided with customization option. This allows custom designing using the [groupTemplate](#) property for both inline and fixed headers.

Grouping with CheckBox

Previously, there is no checkbox for group headers. Now, this feature allow to render checkbox in group header to select the group items in single selection. You can enable this feature by setting [enableGroupCheckBox](#) property value as **true** and **mode** property as **CheckBox**.

Inject the **CheckBoxSelection** module in the MultiSelect to use the checkbox.

INDEX.TS

```

import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
MultiSelect.Inject(CheckBoxSelection);
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
    { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
    { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },

```

```

        { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3'
    },
    { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
    { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },
    { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
    { vegetable: 'Green bean', category: 'Beans', id: 'item7' },
    { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
    { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
    { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
    { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
];
//initiate the MultiSelect
let msObject: MultiSelect = new MultiSelect({
    // bind the sports Data to datasource property
    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { groupBy: 'category', text: 'vegetable', value: 'id' },
    //set the placeholder to MultiSelect input
    placeholder:"Select vegetables",
    // set the type of mode for checkbox to visualized the checkbox added in
    // li element.
    mode: 'CheckBox',
    // set true for enable the selectAll support.
    showSelectAll: true,
    // set value for allowFiltering as true
    allowFiltering: true,
    // set placeholder to filterbar
    filterBarPlaceholder: "Search Vegetables",
    // set true for enableGroupCheckBox property
    enableGroupCheckBox: true
});
//render the component
msObject.appendTo('#select');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <!--element which is going to render the MultiSelect-->
        <input type="text" tabindex="1" id="select">
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Group Template support to MultiSelect.](#)

Filtering in EJ2 JavaScript Multi select control

The MultiSelect has built-in support to filter data items when [allowFiltering](#) is enabled. The filter operation starts as soon as you start typing characters in the MultiSelect input.

To display filtered items in the popup, filter the required data and return it to the MultiSelect via [updateData](#) method by using the [filtering](#) event.

The following sample illustrates how to query the data source and pass the data to the MultiSelect through the [updateData](#) method in [filtering](#) event.

INDEX.TS

```

import { MultiSelect, FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { DataManager, Query } from '@syncfusion/ej2-data';
let searchData: { [key: string]: Object; }[] = [
{ index: "s1", country: "Alaska" }, { index: "s2", country: "California" },
{ index: "s3", country: "Florida" }, { index: "s4", country: "Georgia" }
];
let filter: MultiSelect = new MultiSelect({
    dataSource: searchData,
    // map the appropriate column
    fields: { text: "country", value: "index" },
    // set placeholder to MultiSelect input element
    placeholder: "Select countries",
    // set true to allowFiltering for enable filtering supports
    allowFiltering: true,
    // Bind the filter event
    filtering: function (e: FilteringEventArgs) {
        let query = new Query();
        //frame the query based on search string with filter type.
    }
});

```



```

        query = (e.text != "") ? query.where("country", "startswith",
e.text, true) : query;
        //pass the filter data source, filter query to updateData method.
        e.updateData(searchData, query);
    }
});
filter.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 MultiSelect</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <!--element which is going to render the MultiSelect-->
        <input type="text" tabindex="1" id="select">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limit the minimum filter character

When filtering the list items, you can set the limit for character count to raise remote request and fetch filtered data on the MultiSelect. This can be done by manual validation within the filter event handler.

In the following example, the remote request does not fetch the search data until the search key contains three characters.

INDEX.TS

```
import { MultiSelect, FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let searchData: DataManager = new DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let filter: MultiSelect = new MultiSelect({
    dataSource: searchData,
    query: new Query().select(['ContactName', 'CustomerID']).take(7),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // set placeholder to MultiSelect input element
    placeholder: "Select customers",
    //sort the resulted items
    sortOrder: 'Ascending',
    // set true to allowFiltering for enable filtering supports
    allowFiltering: true,
    //bind the filtering event handler
    filtering: (e: FilteringEventArgs) => {
        // load overall data when search key empty.
        if(e.text == '') e.updateData(searchData);
        else{
            // restrict the remote request until search key contains 3
            characters.
            if (e.text.length < 3) { return; }
            let query: Query = new Query().select(['ContactName',
            'CustomerID']);
            query = (e.text !== '') ? query.where('ContactName', 'startswith',
            e.text, true) : query;
            e.updateData(searchData, query);
        }
    }
});
filter.appendTo('#select');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 MultiSelect</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <!--element which is going to render the MultiSelect-->
        <input type="text" tabindex="1" id="select">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Change the filter type

While filtering, you can change the filter type to **contains**, **startsWith**, or **endsWith** for string type within the filter event handler.

In the following examples, data filtering is done with **endsWith** type.

INDEX.TS

```

import { MultiSelect, FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let searchData: DataManager = new DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let filter: MultiSelect = new MultiSelect({
    dataSource: searchData,
    query: new Query().select(['ContactName', 'CustomerID']).take(7),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // set placeholder to MultiSelect input element
    placeholder: "Select names",
    // set true to allowFiltering for enable filtering supports
    allowFiltering: true,
    //set the height of the popup element
    popupHeight: "250px",
    //sort the resulted items
    sortOrder: 'Ascending',
    //bind the filtering event handler
    filtering: (e: FilteringEventArgs) => {
        // load overall data when search key empty.
        if(e.text == '') e.updateData(searchData);
    }
});

```

```

        else{
            let query: Query = new Query().select(['ContactName',
            'CustomerID']);
            // change the type of filtering
            query = (e.text !== '') ? query.where('ContactName', 'endswith',
            e.text, true) : query;
            e.updateData(searchData, query);
        }
    });
    filter.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 MultiSelect</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <!--element which is going to render the MultiSelect-->
        <input type="text" tabindex="1" id="select">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by passing the fourth optional parameter of the `where` clause.

The following example shows how to perform case-sensitive filter.

INDEX.TS

```
import { MultiSelect, FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
let sportsData: { [key: string]: Object }[] = [
  { id: 'game1', sports: 'Badminton' },
  { id: 'game2', sports: 'Football' },
  { id: 'game3', sports: 'Tennis' },
  { id: 'game4', sports: 'Golf' },
  { id: 'game5', sports: 'Hockey' }
];
let filter: MultiSelect = new MultiSelect({
  dataSource: sportsData,
  // map the appropriate column
  fields: { text: 'sports', value: 'id' },
  // set placeholder to MultiSelect input element
  placeholder: "Select names",
  // set true to allow filtering for enable filtering supports
  allowFiltering: true,
  //set the height of the popup element
  popupHeight: "250px",
  //sort the resulted items
  sortOrder: 'Ascending',
  //bind the filtering event handler
  filtering: (e: FilteringEventArgs) => {
    // load overall data when search key empty.
    if(e.text == '') e.updateData(sportsData);
    else{
      let query: Query = new Query().select(['sports', 'id']);
      //enable the case sensitive filtering by passing false to 4th
      query = (e.text !== '') ? query.where('sports', 'startswith',
        e.text, false) : query;
      e.updateData(sportsData, query);
    }
  }
});
filter.appendTo('#select');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <!--element which is going to render the MultiSelect-->
        <input type="text" tabindex="1" id="select">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Diacritics Filtering

The MultiSelect supports diacritics filtering which will ignore the [diacritics](#) and makes it easier to filter the results in international characters lists when the [ignoreAccent](#) is enabled.

In the following sample, data with diacritics are bound as dataSource for MultiSelect.

INDEX.TS

```

import { MultiSelect } from '@syncfusion/ej2-dropdowns';
// create local data
let data: string[] = [
    'Aeróbics',
    'Aeróbics en Agua',
    'Aerografía',
    'Aeromodelaje',
    'Águilas',
    'Ajedrez',
    'Ala Delta',
    'Álbumes de Música',
    'Alusivos',
    'Análisis de Escritura a Mano'];
// initialize MultiSelect component
let multiObj: MultiSelect = new MultiSelect({
    //set the local data to dataSource property
    dataSource: data,
    // set the placeholder to MultiSelect input element
    placeholder: 'e.g: aero',
    // enabled the ignoreAccent property for ignore the diacritics
    ignoreAccent: true,
    // set true for enable the filtering support.
    allowFiltering: true
});
multiObj.appendTo('#select');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <!--element which is going to render the MultiSelect-->
    <input type="text" tabindex="1" id="select">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [How to bind the data](#)
- [How to group the data using header](#)
- [How to add custom value to the MultiSelect](#)

Custom value in EJ2 JavaScript Multi select control

The MultiSelect allows user to add a new non-present option to the component value when [allowCustomValue](#) is enabled. while selecting the new custom value [customValueSelection](#) event will be triggered.

The following sample demonstrates configuration of custom value support with the MultiSelect component.

INDEX.TS

```
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
    { id: 'game1', sports: 'Badminton' },
    { id: 'game2', sports: 'Football' },
    { id: 'game3', sports: 'Tennis' }
];
//initiate the MultiSelect
let msObject: MultiSelect = new MultiSelect({
    // bind the sports Data to datasource property
    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { text: 'sports', value: 'id' },
    //set the placeholder to MultiSelect input
    placeholder: "Select games",
    //enable custom Value option with multislect
    allowCustomValue: true
});
// render initialized multiSelect
msObject.appendTo('#select');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <!--element which is going to render the MultiSelect-->
    <input type="text" tabindex="1" id="select">
```



```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Checkbox in EJ2 JavaScript Multi select control

The MultiSelect has built-in support to select multiple values through checkbox, when [mode](#) property set as **CheckBox**.

To use checkbox, inject the **CheckBoxSelection** module in the MultiSelect.

INDEX.TS

```

import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
MultiSelect.Inject(CheckBoxSelection);
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
    { id: 'game1', sports: 'Badminton' },
    { id: 'game2', sports: 'Football' },
    { id: 'game3', sports: 'Tennis' },
    { id: 'game4', sports: 'Golf' },
    { id: 'game5', sports: 'Cricket' },
    { id: 'game6', sports: 'Handball' },
    { id: 'game7', sports: 'Karate' },
    { id: 'game8', sports: 'Fencing' },
    { id: 'game9', sports: 'Boxing' }
];
//initiate the MultiSelect
let msObject: MultiSelect = new MultiSelect({
    // bind the sports Data to datasource property
    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { text: 'sports', value: 'id' },
    //set the placeholder to MultiSelect input
    placeholder: "Select games",
    // set the type of mode for checkbox to visualized the checkbox added in
    // li element.
    mode: 'CheckBox',
    //Bind the filter event
    filtering: function (e: FilteringEventArgs) {
        let query = new Query();
        //frame the query based on search string with filter type.
        query = (e.text != "") ? query.where("country", "startswith",
e.text, true) : query;
        //pass the filter data source, filter query to updateData method.
        e.updateData(searchData, query);
    }
});
//render the component
msObject.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <!--element which is going to render the MultiSelect-->
    <input type="text" tabindex="1" id="select">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Select All

The MultiSelect component has in-built support to select the all list items using **Select All** options in the header.

When the [showSelectAll](#) property is set to true, by default Select All text will show. You can customize the name attribute of the Select All option by using [selectAllText](#).

For the unSelect All option, by default unSelect All text will show. You can customize the name attribute of the unSelect All option by using

[unSelectAllText](#).

INDEX.TS

```

import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
MultiSelect.Inject(CheckBoxSelection);
//define the array of complex data

```

```

let sportsData: { [key: string]: Object }[] = [
  { id: 'game1', sports: 'Badminton' },
  { id: 'game2', sports: 'Football' },
  { id: 'game3', sports: 'Tennis' },
  { id: 'game4', sports: 'Golf' },
  { id: 'game5', sports: 'Cricket' },
  { id: 'game6', sports: 'Handball' },
  { id: 'game7', sports: 'Karate' },
  { id: 'game8', sports: 'Fencing' },
  { id: 'game9', sports: 'Boxing' }
];
//initiate the MultiSelect
let msObject: MultiSelect = new MultiSelect({
  // bind the sports Data to datasource property
  dataSource: sportsData,
  // maps the appropriate column to fields property
  fields: { text: 'sports', value: 'id' },
  //set the placeholder to MultiSelect input
  placeholder:"Select games",
  // set the type of mode for checkbox to visualized the checkbox added in
  // li element.
  mode: 'CheckBox',
  // set true for enable the selectAll support.
  showSelectAll: true,
  // set the select all text to MultiSelect checkbox label.
  selectAllText: "Select All"
});
//render the component
msObject.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

```

```

<div id="container" style="margin:0 auto; width:250px;">
  <br>
  <!--element which is going to render the MultiSelect-->
  <input type="text" tabindex="1" id="select">
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection Limit

Defines the limit of the selected items using [maximumSelectionLength](#).

INDEX.TS

```

import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
MultiSelect.Inject(CheckBoxSelection);
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
  { id: 'game1', sports: 'Badminton' },
  { id: 'game2', sports: 'Football' },
  { id: 'game3', sports: 'Tennis' },
  { id: 'game4', sports: 'Golf' },
  { id: 'game5', sports: 'Cricket' },
  { id: 'game6', sports: 'Handball' },
  { id: 'game7', sports: 'Karate' },
  { id: 'game8', sports: 'Fencing' },
  { id: 'game9', sports: 'Boxing' }
];
//initiate the MultiSelect
let msObject: MultiSelect = new MultiSelect({
  // bind the sports Data to datasource property
  dataSource: sportsData,
  // maps the appropriate column to fields property
  fields: { text: 'sports', value: 'id' },
  //set the placeholder to MultiSelect input
  placeholder:"Select games",
  // set the type of mode for checkbox to visualized the checkbox added in
  // li element.
  mode: 'CheckBox',
  // Sets limitation to the value selection
  maximumSelectionLength: 3
});
//render the component
msObject.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <!--element which is going to render the MultiSelect-->
        <input type="text" tabindex="1" id="select">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection Reordering

Using [enableSelectionOrder](#) to Reorder the selected items in popup visibility state.

INDEX.TS

```

import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
MultiSelect.Inject(CheckBoxSelection);
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
    { id: 'game1', sports: 'Badminton' },
    { id: 'game2', sports: 'Football' },
    { id: 'game3', sports: 'Tennis' },
    { id: 'game4', sports: 'Golf' },
    { id: 'game5', sports: 'Cricket' },
    { id: 'game6', sports: 'Handball' },
    { id: 'game7', sports: 'Karate' },
    { id: 'game8', sports: 'Fencing' },
    { id: 'game9', sports: 'Boxing' }
];
//initiate the MultiSelect
let msObject: MultiSelect = new MultiSelect({
    // bind the sports Data to datasource property

```

```

    dataSource: sportsData,
    // maps the appropriate column to fields property
    fields: { text: 'sports', value: 'id' },
    //set the placeholder to MultiSelect input
    placeholder:"Select games",
    // set the type of mode for checkbox to visualized the checkbox added in
    li element.
    mode: 'CheckBox',
    // Reorder the selected items in popup visibility state.
    enableSelectionOrder: false
  });
  //render the component
  msObject.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <!--element which is going to render the MultiSelect-->
    <input type="text" tabindex="1" id="select">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to bind the data](#)
- [How to filter the bound data](#)
- [How to add custom value to the MultiSelect](#)
- [How to render grouping with checkbox.](#)

Chip customization in EJ2 JavaScript Multi select control

The MultiSelect allows the user to customize the selected chip element through the [tagging](#) event. In that event, you can set the custom classes to chip element via that event argument of `setClass` method.

The following sample demonstrates chip-customization with the MultiSelect component.

INDEX.TS

```
import { MultiSelect, TaggingEventArgs } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let colorsData: { [key: string]: Object; }[] = [
    { Color: 'Chocolate', Code: '#75523C' },
    { Color: 'CadetBlue', Code: '#3B8289' },
    { Color: 'DarkOrange', Code: '#FF843D' },
    { Color: 'DarkRed', Code: '#CA3832' },
    { Color: 'Fuchsia', Code: '#D44FA3' },
    { Color: 'HotPink', Code: '#F23F82' },
    { Color: 'Indigo', Code: '#2F5D81' },
    { Color: 'LimeGreen', Code: '#4CD242' },
    { Color: 'OrangeRed', Code: '#FE2A00' },
    { Color: 'Tomato', Code: '#FF745C' }
];
//initiate the MultiSelect
let msObject: MultiSelect = new MultiSelect({
    // bind the sports Data to datasource property
    dataSource: colorsData,
    // maps the appropriate column to fields property
    fields: { text: 'Color', value: 'Code' },
    // set the value to MultiSelect
    value: ['#75523C', '#4CD242', '#FF745C'],
    //set the placeholder to MultiSelect input
    placeholder: "Select a color",
    // set the type of mode for how to visualized the selected items in
    // input element.
    mode: 'Box',
    // bind the tagging event
    tagging: (e: TaggingEventArgs) => {
        // set the current selected item text as class to chip element.
        e.setClass((e.itemData as
any) [msObject.fields.text].toLowerCase());
    }
});
// render initialized multiSelect
msObject.appendTo('#select');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 MultiSelect</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <!--element which is going to render the MultiSelect-->
        <input type="text" tabindex="1" id="select">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Localization in EJ2 JavaScript Multi select control

The Localization library allows you to localize static text content of the [noRecordsTemplate](#) and [actionFailureTemplate](#) properties according to the culture currently assigned to the MultiSelect.

| Locale key | en-US (default) |

|-----|-----|

| noRecordsTemplate | No records found |

| actionFailureTemplate | The request failed |

Loading translations

To load translation object to your application, use load function of the **L10n** class.

In the following sample, French culture is set to the MultiSelect and no data is loaded. Hence, the [noRecordsTemplate](#) property displays its text in French culture initially, and if the sample is run offline, the [actionFailureTemplate](#) property displays its text appropriately.

INDEX.TS

```
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
// import L10n class for load function
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
// bind remotedata to showcase actionFailureTemplate in offline.
let customerData: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let msObj: MultiSelect = new MultiSelect({
    dataSource: customerData,
    // set locale culture to MultiSelect
    locale: 'fr-BE',
    // map appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // take 0 item to showcase noRecordsTemplate property.
    query: new Query().select(['ContactName', 'CustomerID']).take(0),
    // set placeholder to MultiSelect input element
    placeholder: 'Sélectionnez un éléments'
});
msObj.appendTo('#select');
L10n.load({
    'fr-BE': {
        'multi-select': {
            'noRecordsTemplate': "Aucun enregistrement trouvé",
            'actionFailureTemplate': "Modèle d'échec d'action"
        }
    }
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
```

```

</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <br>
        <!--element which is going to render the MultiSelect-->
        <input type="text" tabindex="1" id="select">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Accessibility](#)
- [How to bind the data to the combobox](#)

Style in EJ2 JavaScript Multi select control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the background color of wrapper element

Use the following CSS to customize the background color of wrapper element.

```

.e-multiselect.e-input-group .e-multi-select-wrapper {
background-color: red;
}

```

Customizing the appearance of the delimiter wrapper element

Use the following CSS to customize the appearance of delimiter wrapper element.

```

.e-multiselect .e-delim-values {
-webkit-text-fill-color: blue;
font-size: 16px;
font-family: cursive;
}

```

Customizing the appearance of chips

Use the following CSS to customize the appearance of selected chips.

```
,  
  
.e-multiselect .e-multi-select-wrapper .e-chips .e-chipcontent {  
font-family: cursive;  
font-size: 20px;  
-webkit-text-fill-color: blue;  
}  
  
.e-multi-select-wrapper .e-chips {  
background-color: aqua;  
height: 26px;  
}  
,
```

Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

```
,  
  
.e-multiselect.e-input-group .e-input-group-icon, .e-multiselect.e-input-group.e-control-wrapper .e-input-group-icon:hover {  
color: red;  
font-size: 14px;  
}  
,
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
,  
  
.e-multiselect.e-input-group.e-control-wrapper.e-input-focus::before, .e-multiselect.e-input-group.e-control-wrapper.e-input-focus::after {  
background: #c000ff;  
}  
,
```

Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

```
,  
  
.e-multiselect.e-disabled .e-multi-select-wrapper .e-delim-values {  
-webkit-text-fill-color: red;  
}
```

```
}
,
```

Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```
,
```

```
.e-multiselect input.e-dropdownbase::placeholder {
color: red;
}
```

```
,
```

Customizing the placeholder to add mandatory indicator(*)

Use the following CSS to add the mandatory indicator * to the float label element.

```
,
```

```
.e-input-group.e-control-wrapper.e-float-input .e-float-text::after {
content: "*";
color: red;
}
```

```
,
```

Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
,
```

```
.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-
wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-
float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left)
.e-float-line::after {
background-color: #2319b8;
}
```

```
.e-multiselect.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top, .e-
float-input.e-control-wrapper:not(.e-error).e-input-focus input ~ label.e-float-text {
color: #2319b8;
}
```

```
,
```

Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
,
```

```
.e-outline.e-input-group.e-input-focus: hover: not(.e-success): not(.e-warning): not(.e-error): not(.e-
disabled): not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper: hover: not(.e-
```

```

success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-
input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled),.e-outline.e-input-group.e-
control-wrapper.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled) {
border-color: #b1bd15;
box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;
}

```

Customizing the background color of focus, hover, and active item's

Use the following CSS to customize the background color of focus, hover and active item's.

```

.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-
list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
background-color: #1f9c99;
color: #2319b8;
}

```

Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

```

.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px
}

```

Customizing the color of the checkbox

Use the following CSS to customize the color of checkbox.

```

.e-popup .e-checkbox-wrapper .e-frame.e-check, .e-popup .e-checkbox-wrapper:hover .e-frame.e-check
{
background-color: green;
color: white;
}

```

Accessibility in EJ2 JavaScript Multi select control

The MultiSelect component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties along with keyboard support. This component is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The MultiSelect component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the MultiSelect component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The MultiSelect component uses the **Listbox** role, and each list item has an **option** role. The following **ARIA attributes** denote the MultiSelect state.

| Properties | Functionalities |

| --- | --- |

| aria-haspopup | Indicates whether the MultiSelect input element has a popup list or not. |

| aria-expanded | Indicates whether the popup list has expanded or not. |

| aria-selected | Indicates the selected option. |

| aria-readonly | Indicates the readonly state of the MultiSelect element. |

| aria-disabled | Indicates whether the MultiSelect component is in a disabled state or not. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

Keyboard interaction

You can use the following key shortcuts to access the MultiSelect without interruptions.

| Keyboard shortcuts | Actions |

| --- | --- |

| Arrow Down | Set focus at the first item in the MultiSelect when no item selected. Otherwise, moves focus next to the currently selected item. |

| Arrow Up | Moves focus previous to the currently selected one. |

| Page Down | Scrolls down to the next page and set focus to the first item when popup list opens. |

| Page Up | Scrolls up to the previous page and set focus to the first item when popup list opens. |

| Enter | Selects the focused item, and popup list closes when it is in open state. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Shift + tab | Focuses on the previous TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Alt + Down | Opens the popup list. |

| Alt + Up | Closes the popup list. |

| Esc(Escape) | Closes the popup list when it is in an open state and the currently selected item remains the same. |

| Home | set focus to the first item. |

| End | set focus to the last item. |

In the below sample, focus the MultiSelect component using alt+t keys.

INDEX.TS

```
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
// defined the array of data
let gameList: { [key: string]: Object }[] = [
    { id: 'Game1', game: 'Badminton' },
    { id: 'Game2', game: 'Basketball' },
    { id: 'Game3', game: 'Cricket' },
    { id: 'Game4', game: 'Football' },
    { id: 'Game5', game: 'Golf' },
    { id: 'Game6', game: 'Hockey' },
    { id: 'Game7', game: 'Rugby' },
    { id: 'Game8', game: 'Snooker' },
    { id: 'Game9', game: 'Tennis' },
];
// initialize MultiSelect component
let msObject: MultiSelect = new MultiSelect({
    //set the data to dataSource property
    dataSource: gameList,
    //map to column to fields
    fields: { text: 'game', value: 'id' },
    // set placeholder to MultiSelect input element
    placeholder: "Select games",
    // set the popup list height
    popupHeight: '200px'
});
// render initialized MultiSelect
msObject.appendTo('#select');
document.onkeyup = function (e) {
    if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        msObject.focusIn();
    }
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 MultiSelect</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="select">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Ensuring accessibility

The MultiSelect component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the MultiSelect component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the MultiSelect component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

How To

Icons support in EJ2 JavaScript Multi select control

You can render **icons** to the list items by mapping the [iconCss](#) field. This [iconCss](#) field create a span in the list item with mapped class name to allow styling as per your need.

In the following sample, icon classes are mapped with [iconCss](#) field.

INDEX.TS

```

import { MultiSelect } from '@syncfusion/ej2-dropdowns';
let sortFormatData: { [key: string]: Object }[] = [
    { class: 'asc-sort', type: 'Sort A to Z', id: '1' },
    { class: 'dsc-sort', type: 'Sort Z to A ', id: '2' },
    { class: 'filter', type: 'Filter', id: '3' },
    { class: 'clear', type: 'Clear', id: '4' }
];
let sortFormat: MultiSelect = new MultiSelect({

```

```

    dataSource: sortFormatData,
    // map the icon column to iconCSS field.
    fields: { text: 'type', iconCss: 'class', value: 'id' },
    placeholder: 'Select a format'
  });
  sortFormat.appendTo('#select');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 MultiSelect</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin:0 auto; width:250px;">
    <br>
    <input type="text" tabindex="1" id="select">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Achieve virtual scrolling in EJ2 JavaScript Multi select control

The Virtual Scrolling is used to display a large amount of data without buffering the entire load of a huge database record in the MultiSelect, that is, when scrolling, the request is sent and fetch some amount of data from the server dynamically. Using the `scroll` event, get the data and generate the list add to popup using the `addItem` method.

Refer to the following code sample for virtual scrolling.

INDEX.TS

```

import { MultiSelect } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/',
    crossDomain: true
});

// initialize MultiSelect component
let mulObj: MultiSelect = new MultiSelect({
    // bind the DataManager instance to dataSource property
    dataSource: data,
    // bind the Query instance to query property
    query: new Query().from('Customers').select('ContactName').take(7),
    // map the appropriate columns to fields property
    fields: { text: 'ContactName', value: 'ContactName' },
    // set the placeholder to MultiSelect input element
    placeholder: 'Select a customer',
    // sort the resulted items
    sortOrder: 'Ascending',
    // set the height of the popup element
    popupHeight: '200px',
    actionComplete: function (e: any) {
        let operator: Query = new
Query().from('Customers').select('ContactName');
        let start: number = 7;
        let end: number = 12;
        let listElement: HTMLElement = this.list;
        listElement.addEventListener('scroll', () => {
            if ((listElement.scrollTop + listElement.offsetHeight >=
listElement.scrollHeight)) {
                let filterQuery = operator.clone();
                data.executeQuery(filterQuery.range(start,
end)).then((event: any) => {
                    start = end;
                    end += 5;
                    mulObj.addItem(event.result as { [key: string]: Object
}[]);
                }).catch((e: Object) => {
                });
            }
        });
    }
});
mulObj.appendTo('#atcelement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 AutoComplete</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <input type="text" tabindex="1" id="atcelement">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Validation in EJ2 JavaScript Multi select control

MultiSelect component inside form can be validated through **FormValidator**. Add the name attribute of component to be validated inside rules of FormValidator. Error message after validation can be placed in DOM based on the requirement through customPlacement.

In the following sample, validation is added for MultiSelect component.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
let listObj1: MultiSelect = new MultiSelect({
    // set the placeholder to MultiSelect input element
    placeholder: 'Favorite Sports',
    // set the type of mode for how to visualized the selected items in
    input element.
    mode: 'Default'
});
listObj1.appendTo('#default');
// Initialize Submit button
let buttonFormValidate: Button = new Button({ isPrimary: true });
buttonFormValidate.appendTo('#validateSubmit');
// Initialize Reset button
let buttonReset: Button = new Button({});
buttonReset.appendTo('#resetbtn');
// Initialize Custom placement
let option: FormValidatorModel = {
    rules: {

```

```

        // Initialize the CustomPlacement.
        default: { required: true }
    },
    customPlacement: (inputElement: HTMLElement, error: HTMLElement)=>{
inputElement.parentElement.parentElement.parentElement.insertBefore(error,
inputElement.parentElement.parentElement.nextSibling);
    }
    };
    // Initialize Form validation
    let formObj: FormValidator;
    formObj = new FormValidator('#formId', option);
    let formId: HTMLElement =
<HTMLElement>document.getElementById('formId');
    document.getElementById('formId').addEventListener(
        'submit',
        (e: Event) => {
            e.preventDefault();
            if (formObj.validate()) {
                alert('Success');
                formObj.reset();
            }
        }
    );
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 MultiSelect</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin:0 auto; width:250px;">
        <div class="col-lg-12 control-section">
            <div class="content-wrapper" style="margin-bottom: 25px;">
                <div class="form-title"><span>Add Customer Details</span></div>
                <form id="formId" class="form-horizontal" novalidate="">
                    <div class="form-group">

```

```

        <div class="e-float-input">
        <select id="default" name="default">
        <option value="Game1">American Football</option>
        <option value="Game2">Badminton</option>
        <option value="Game3">Basketball</option>
        <option value="Game4">Cricket</option>
        <option value="Game5">Football</option>
        <option value="Game6">Golf</option>
        <option value="Game7">Hockey</option>
        <option value="Game8">Rugby</option>
        <option value="Game9">Snooker</option>
        <option value="Game10">Tennis</option>
        </select>
        </div>
        <div id="userError"></div>
    </div>
    <div class="row">
        <div style="width: 320px;margin:0px auto;height:
100px;padding-top: 25px;">
            <div style="display: inline-block;">
                <button id="validateSubmit" class="samplebtn e-
control e-btn e-primary" type="submit" style="height:40px;width: 150px;"
data-ripple="true">Submit</button>
            </div>
            <div style="float: right;">
                <button id="resetbtn" class="samplebtn e-control e-
btn" type="reset" style="height:40px;width: 150px;" data-
ripple="true">Clear</button>
            </div>
        </div>
    </div>
</form>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

NumericTextBox

Formats in EJ2 JavaScript Numerictextbox control

You can format the value of NumericTextBox using [format](#) property. The value will be displayed in the specified format when the component is in focused out state. The format string supports both the [standard numeric format string](#) and [custom numeric format string](#).

Standard formats

From the [standard numeric formats](#), you can use the numeric related format specifiers such as **n**, **p** and **c** in the NumericTextBox component. By using these format specifiers, you can achieve the percentage and currency textbox behavior also.

The below example demonstrates percentage and currency formats.

INDEX.TS

```
import {NumericTextBox} from '@syncfusion/ej2-inputs';
/* 'p' specifier */
let percent: NumericTextBox = new NumericTextBox({
    // sets percentage with 2 numbers of decimal places format
    format: 'p2',
    value: 0.5,
    min: 0,
    max: 1,
    step: 0.01,
    placeholder: 'Percentage format',
    floatLabelType: 'Auto'
});
percent.appendTo('#percent');
/* 'c' specifier */
let currency: NumericTextBox = new NumericTextBox({
    // sets currency with 2 numbers of decimal places format
    format: 'c2'
    value: 10,
    placeholder: 'Currency format',
    floatLabelType: 'Auto'
});
currency.appendTo('#currency');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 NumericTextBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript NumericTextBox Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
```

```

        <input id="percent" type="text">
    </div>
    <div class="wrap">
        <input id="currency" type="text">
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 35px auto;
    width: 240px;
    padding-top: 30px;
}

```

Custom formats

From the [custom numeric format string](#), you can provide any custom format by combining one or more custom specifiers.

The below examples demonstrate format the value by using currency format string **#** and **0**.

INDEX.TS

```

import {NumericTextBox} from '@syncfusion/ej2-inputs';
/* '#' specifier */
let numeric = NumericTextBox = new NumericTextBox({
    // sets the format using custom format string '#'
    format: '###.##',
    value: 10,
    placeholder: 'Custom format string #',
    floatLabelType: 'Auto'
});
numeric.appendTo('#numeric');
/* '0' specifier */
let numeric1: NumericTextBox = new NumericTextBox({

```



```
// sets the format using custom format string `0`
format: '000.00',
value: 10,
placeholder: 'Custom format string 0',
floatLabelType: 'Auto'
});
numeric1.appendTo('#numeric1');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 NumericTextBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript NumericTextBox Component
with Custom Format">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="numeric" type="text">
    </div>
    <div class="wrap">
      <input id="numeric1" type="text">
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue','calibiri';
```

```

font-size: 14px;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.wrap {
margin: 35px auto;
width: 240px;
padding-top: 30px;
}

```

Globalization in EJ2 JavaScript Numerictextbox control

Localization

[Localization](#) library allows users to localize the default text contents of the NumericTextBox to different cultures using the [locale](#) property. In NumericTextBox, spin buttons title for the tooltip will be localized based on the culture.

| Locale key | en-US (default) |

|-----|-----|

| incrementTitle | Increment value |

| decrementTitle | Decrement value |

Loading translations

To load translation object in your application use `load` function of `L10n` class.

The below example demonstrates the NumericTextBox in **German** culture with the spin buttons tooltip.

INDEX.TS

```

import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { L10n } from '@syncfusion/ej2-base';
// Load `German` culture to override spin buttons tooltip text
L10n.load({
  'de': {
    'numerictextbox': {
      incrementTitle: 'Wert erhöhen', decrementTitle: 'Dekrementwert'
    }
  },
});
let numeric: NumericTextBox = new NumericTextBox({
  // sets `German` culture using the culture value 'de'
  locale: 'de',
  // sets value to the NumericTextBox
  value: 10
});
// renders initialized NumericTextBox
numeric.appendTo('#numeric');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```
<title>EJ2 NumericTextBox</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="TypeScript NumericTextBox Component">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="numeric" type="text">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 240px;
    padding-top: 100px;
}
```

Internationalization

Internationalization library provides support for formatting and parsing the number by using the official [Unicode CLDR](#) JSON data and also provides the `loadCldr` method to load the culture specific CLDR JSON data. The NumericTextBox comes with built-in

internationalization support to adapt based on culture. For more information about internationalization, refer to this [link](#).

By default, all the Essential JS 2 component are specific to English culture ('en-US'). If you want to go with the different culture other than `English`, follow the below steps.

- Install the `CLDR-Data` package by using the below command (it installs the CLDR JSON data). For more information about CLDR-Data, refer to this [link](#).

```
npm install cldr-data --save
```

Once the package installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

- Now import the installed CLDR JSON data into the `app.ts` file. To import JSON data we need to install the JSON plugin loader. Here we have used the SystemJS JSON plugin loader.

```
npm install systemjs-plugin-json --save-dev
```

- Once installed, configure the `system.config.js` configuration settings as like below to map the `systemjs-plugin-json` loader.

```
`ts
System.config({
  paths: {
    'syncfusion:': 'npm:@syncfusion/'
  },
  map: {
    app: 'app',
    //Syncfusion packages mapping
    '@syncfusion/ej2-base': 'syncfusion:ej2-base/dist/ej2-base.umd.min.js',
    '@syncfusion/ej2-inputs': 'syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js',
    'cldr-data': 'npm:cldr-data',
```

```
"plugin-json": "npm:systemjs-plugin-json/json.js"
},
meta: {
  '*.json': { loader: 'plugin-json' }
},
packages: {
  'app': { main: 'app', defaultExtension: 'js' },
  'cldr-data': { main: 'index.js', defaultExtension: 'js' }
}
});
System.import('app');
`
```

- Now import the required culture from the installed location to `app.ts` file as like the below code snippets.

```
`ts
declare var require: any;
loadCldr(
  require('cldr-data/main/de/numbers.json'),
  require('cldr-data/main/de/currencies.json'),
  require('cldr-data/supplemental/numberingSystems.json'),
  require('cldr-data/supplemental/currencyData.json')
);
`
```

- Set the culture by using the [locale](#) property.

The below example demonstrates the NumericTextBox in `German` culture with the `EUR` currency format.

INDEX.TS

```
import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as currencyData from './currencyData.json';
import * as numbers from './numbers.json';
import * as currencies from './currencies.json';
loadCldr(numberingSystems, currencyData, numbers, currencies);
```

```

L10n.load({
  'de': {
    'numerictextbox': { incrementTitle: 'Wert erhöhen', decrementTitle:
'Dekrementwert' }
  }
});
let currency: NumericTextBox = new NumericTextBox({
  // sets `German` culture using the culture value 'de'
  locale: 'de',
  // sets the 'EUR' currency format
  currency: 'EUR',
  format: 'c2',
  value: 100,
});
currency.appendTo('#numeric');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 NumericTextBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript NumericTextBox Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="numeric" type="text">
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}

```

```

}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue','calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  margin: 0 auto;
  width: 240px;
  padding-top: 100px;
}

```

Right to Left(RTL)

RTL provides an option to switch the text direction and layout of the NumericTextBox component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL NumericTextBox, set the [enableRtl](#) to true.

INDEX.TS

```

import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { loadCldr,L10n } from '@syncfusion/ej2-base';
L10n.load({
  'ar-AE': {
    'numerictextbox': { incrementTitle: 'قيمة الزيادة', decrementTitle:
'قيمة تناقص' }
  }
});
let currency: NumericTextBox = new NumericTextBox({
  // sets `Arabic` culture using the culture value 'ar-AE'
  locale: 'ar-AE',
  value: 100,
  placeholder: 'أدخل القيمة',
  enableRtl: true,
  floatLabelType: 'Auto'
});
currency.appendTo('#numeric');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 NumericTextBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript NumericTextBox Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="numeric" type="text">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 240px;
    padding-top: 100px;
}

```

Accessibility in EJ2 JavaScript Numerictextbox control

The Numerictextbox component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Numerictextbox component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The NumericTextBox characterized with complete ARIA Accessibility support which helps to accessible by on-screen readers and other assistive technology devices. This component designed with the reference of the guidelines document given in [WAI ARAI Accessibility practices](#).

The NumericTextBox uses the `spinbutton` role and following ARIA properties to its element based on its state.

| Property | Functionality |

| --- | --- |

| aria-live | The `aria-live` attribute indicates the priority of updates to a live region |

| **aria-valuemin** | The **aria-valuemin** property specifies the minimum allowable range of the NumericTextBox. |

| **aria-valuemax** | The **aria-valuemax** property specifies the maximum allowable range of the NumericTextBox. |

| **aria-disabled** | The **aria-disabled** property indicates disabled state of the NumericTextBox. |

| **aria-readonly** | The **aria-readonly** property indicates the read-only state of the NumericTextBox. |

| **aria-valuenow** | The **aria-valuenow** property specifies the current value of the NumericTextBox. |

| **aria-invalid** | The **aria-invalid** property indicates that the user input is incorrect or not within acceptable ranges. |

| **aria-label** | The **aria-label** property indicates a string value that labels the NumericTextBox. |

Keyboard interaction

Keyboard interaction of the NumericTextBox component has been designed based on

[WAI-ARIA Practices](#) described for the NumericTextBox and

it is an alternative to mouse actions to interact with the NumericTextBox.

The below table shows shortcut keys and its corresponding usage.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| **Arrow Down** | **Increments the value.** |

| **Arrow Up** | **Decrements the value** |

INDEX.TS

```
import {NumericTextBox} from '@syncfusion/ej2-inputs';
let numeric: NumericTextBox = new NumericTextBox({
    // sets value to the NumericTextBox
    value: 10
});
numeric.appendTo('#numeric');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 NumericTextBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript NumericTextBox Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="numeric" type="text">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 240px;
    padding-top:100px;
}
```

Ensuring accessibility

The NumericTextBox component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the NumericTextBox component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the NumericTextBox component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Style appearance in EJ2 JavaScript Numerictextbox control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of NumericTextBox wrapper element

Use the following CSS to customize the appearance of wrapper element.

/ To specify height and font size /

```
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input, .e-input-group textarea.e-input, .e-input-group.e-control-wrapper textarea.e-input {
```

```
height: 40px;
```

```
font-size: 20px;
```

```
}
```

Customizing the NumericTextBox icons

Use the following CSS to customize the Numeric TextBox icons

/ To specify font size and background color /

```
.e-numeric.e-control-wrapper.e-input-group .e-input-group-icon {
```

```
font-size: 20px;
```

```
background-color: beige;
```

```
}
```

How To

Customize the ui appearance of the control in EJ2 JavaScript Numerictextbox control

You can change the appearance of the NumericTextBox by adding custom `cssClass` to the component and enabling styles. Refer to the following example to change the appearance of the NumericTextBox.

INDEX.TS

```
import { NumericTextBox } from '@syncfusion/ej2-inputs';
// initializes the NumericTextBox component
let numeric: NumericTextBox = new NumericTextBox({
    // sets value to the NumericTextBox
    value: 10,
    placeholder: 'Enter value ',
    floatLabelType: 'Always',
    //adding custom css class to NumericTextBox
    cssClass: 'e-style'
});
// renders initialized NumericTextBox
numeric.appendTo('#numeric1');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 NumericTextBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript NumericTextBox Component
with Custom cssClass">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="numeric1" type="text" name="numericValue/">
    </div>
  </div>

<style>
.e-numeric.e-style .e-control.e-numerictextbox {
  color: royalblue ;
  font-size: xx-large ;
  border: 0px ;
}
.e-input-group.e-control-wrapper:not(.e-success):not(.e-warning):not(.e-
error):not(.e-float-icon-left), .e-float-input.e-control-
wrapper:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-
disabled):not(.e-float-icon-left) {
  border-color: royalblue;
}
.e-control-wrapper.e-numeric.e-float-input.e-style .e-spin-down {
  color:royalblue;
}
.e-control-wrapper.e-numeric.e-float-input.e-style .e-float-line::before {
  background: royalblue ;
}
.e-control-wrapper.e-numeric.e-float-input.e-style .e-float-line::after {
  background: royalblue ;
}
.e-control-wrapper.e-numeric.e-float-input.e-style .e-spin-up {
  color:royalblue ;
}
.e-control-wrapper.e-numeric.e-float-input.e-style .e-float-text.e-label-
top {

```

```

        color: royalblue ;
        font-size: medium ;
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008c00;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 240px;
    padding-top: 100px;
}

```

Customize the spin buttons up and down arrow in EJ2 JavaScript Numerictextbox control

This section explains about how to change/customize spin up and down icons. You can customize spin button icons using **e-spin-up** and **e-spin-down** classes of those buttons.

You can override the default icons of **e-spin-up** and **e-spin-down** classes using the following CSS code snippets.

```

.e-numeric .e-input-group-icon.e-spin-up:before {
content: "\e823";
color: rgba(0, 0, 0, 0.54);
}
.e-numeric .e-input-group-icon.e-spin-down:before {
content: "\e934";
color: rgba(0, 0, 0, 0.54);
}

```

```
}  
,
```

INDEX.TS

```
import {NumericTextBox} from '@syncfusion/ej2-inputs';  
// initializes NumericTextBox component  
let numeric: NumericTextBox = new NumericTextBox({  
    // sets value to the NumericTextBox  
    value: 10  
});  
// renders initialized NumericTextBox  
numeric.appendTo('#numeric');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
    <title>EJ2 NumericTextBox</title>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <meta name="description" content="TypeScript NumericTextBox Component  
with Custom Icons">  
    <meta name="author" content="Syncfusion">  
    <link href="styles.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
base/styles/material.css" rel="stylesheet">  
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
inputs/styles/material.css" rel="stylesheet">  
  
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"  
type="text/javascript"></script>  
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type  
="text/javascript"></script>  
</head>  
<body>  
  
    <div id="container">  
        <div class="wrap">  
            <input id="numeric" type="text">  
        </div>  
    </div>  
<script>  
var ele = document.getElementById('container');  
if(ele) {  
    ele.style.visibility = "visible";  
}  
    </script>  
<script src="index.js" type="text/javascript"></script>  
</body></html>
```

STYLES.CSS

```
#container {  
    visibility: hidden;
```

```

}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue','calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  margin: 0 auto;
  width: 240px;
  padding-top:100px;
}
/* csslint ignore:start */
.e-numeric .e-input-group-icon.e-spin-up:before {
  content: "\e823";
  color: rgba(0, 0, 0, 0.54);
}
.e-numeric .e-input-group-icon.e-spin-down:before {
  content: "\e934";
  color: rgba(0, 0, 0, 0.54);
}
/* csslint ignore:end */

```

Customize the step value and hide spin buttons in EJ2 JavaScript Numerictextbox control

The spin buttons allow you to increase or decrease the value with the predefined [step](#) value. The visibility of spin buttons can be set using the [showSpinButton](#) property.

INDEX.TS

```

import {NumericTextBox} from '@syncfusion/ej2-inputs';
let hideButtons: NumericTextBox = new NumericTextBox({
  // sets the step value as '2' to increase/decrease the value by '2'
  step: 2,
  // sets the showSpinButton value as `false` to hide the spin buttons
  showSpinButton: false
  min: 10,
  max: 100,
  value: 16
});
hideButtons.appendTo('#numeric');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 NumericTextBox</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript NumericTextBox Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="numeric" type="text">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 240px;
    padding-top: 100px;
}

```

Maintain trailing zeros in numerictextbox in EJ2 JavaScript Numerictextbox control

By default, trailing zeros disappear when the NumericTextBox gets focus. However, you can use the following sample to maintain the trailing zeros while focusing the NumericTextBox.

INDEX.TS

```
import {NumericTextBox} from '@syncfusion/ej2-inputs';
```

```
//maintains trailing zeros while focusing
let numericFocus: function () {
    var numericObj = this.ej2_instances ? this.ej2_instances[0] : this;
    numericObj.element.value =
    numericObj.formattedValue(numericObj.decimals, +numericObj.element.value);
}
// Render the Numeric Textbox
let numeric: NumericTextBox = new NumericTextBox({
    value: 10,
    decimals: 2,
    format: 'n2',
    placeholder: 'NumericTextbox',
    floatLabelType: 'Always' ,
    change: numericFocus
});
numeric.appendTo('#numeric');
document.getElementById('numeric').addEventListener('focus', numericFocus);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 NumericTextBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript NumericTextBox Component to
maintain trailing zeros">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="numeric" type="text">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 35px auto;
    width: 240px;
    padding-top: 100px;
}
```

Perform custom validation using form validator in EJ2 JavaScript Numerictextbox control

This section explains how to perform custom validation on the NumericTextBox using FormValidator.

The NumericTextBox will be validated when the value changes or the user clicks the submit button.

Validation can be performed by adding custom validation in the rules collection of the FormValidator.

INDEX.TS

```
import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
// initializes the NumericTextBox component
let numeric: NumericTextBox = new NumericTextBox({
    min: 10,
    max: 100,
    strictMode : false,
    placeholder: 'NumericTextbox',
    floatLabelType: 'Always',
    change: function(args) {
        if (numeric.value != null)
            formObject.validate("numericRange");
    }
});
numeric.appendTo('#numeric1');
let button: Button = new Button();
button.appendTo('#submit_btn');
// checks the value of NumericTextbox and returns the corresponding boolean value
let customFn: (args: { [key: string]: string }) => boolean = (args: { [key: string]: string }) => {
    if(numeric.value>=10 && numeric.value<=100) {
        return true;
    }
    else {
        return false;
    }
}
```

```

};
// sets required property in the FormValidator rules collection
let options: FormValidatorModel = {
    rules: {
        'numericRange': { required: [true, "Number is required"] },
    }
}
// defines FormValidator to validate the NumericTextBox
let formObject: FormValidator = new FormValidator('#form-element', options);
//rules for validating the NumericTextbox
formObject.addRules('numericRange', { range: [customFn, "Please enter a
number between 10 to 100"] });
// places error label outside the NumericTextBox using the customPlacement
event of FormValidator
let customPlace: (element: HTMLElement, error: HTMLElement) => void =
(element: HTMLElement, error: HTMLElement) => {
    element.parentNode.parentNode.appendChild(error);
};
formObject.customPlacement = customPlace;
let submitBtn: HTMLInputElement =
<HTMLInputElement>document.getElementById('submit_btn');
submitBtn.onclick = () => {
    // validates the NumericTextBox
    formObject.validate("numericRange");
    let ele: HTMLInputElement =
<HTMLInputElement>document.getElementById('numeric1');
    // checks for incomplete value and alerts the formt submit
    if (ele.value !== "" && ele.value >=10 && ele.value<=100) {
        alert("Submitted");
    }
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 NumericTextBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript NumericTextBox Component
with Form Validation">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div class="wrap">
    <form id="form-element" class="form-horizontal">
      <div class="form-group">
        <input id="numeric1" type="text" name="numericRange"
class="form-control">
        <div id="error"></div>
        <button type="button" id="submit_btn" style="margin-top:
10px">Submit</button>
      </div>
    </form>
  </div>
</div>
<style>
.e-numeric.e-control-wrapper {
  margin-bottom: 20px;
}
label.e-error {
  margin-top: -50px;
}
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  margin: 0 auto;
  width: 240px;
  padding-top: 100px;
}

```

Prevent nullable input in numerictextbox in EJ2 JavaScript Numerictextbox control

By default, the value of the NumericTextBox sets to null. In some applications, the value of the NumericTextBox should not be null at any instance. In such cases, following sample can be used to prevent nullable input in NumericTextBox.

INDEX.TS

```
import {NumericTextBox} from '@syncfusion/ej2-inputs';
// initializes NumericTextBox component
let numeric: NumericTextBox = new NumericTextBox({
    // sets value to the NumericTextBox
    placeholder: 'NumericTextbox',
    floatLabelType: 'Always' ,
    // prevents nullable value during initialization
    created: function (args) {
        if (this.value==null) {
            this.value=0;
        }
    },
    blur: function (args) {
        // checks for nullable value while focus out
        if (args.value==null)
            numeric1.value=0;
    }
});
// renders initialized NumericTextBox
numeric.appendTo('#numeric1');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 NumericTextBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript NumericTextBox Component to
prevent nullable input">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="numeric1" type="text">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 0 auto;
    width: 240px;
    padding-top: 100px;
}

```

Ej1 api migration in EJ2 JavaScript Numerictextbox control

This article describes the API migration process of NumericTextBox component from Essential JS 1 to Essential JS 2.

Common

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Triggers on creation | **Event:** *create*

\$("#numeric").ejNumericTextBox({
value: 120,
create: function(){}
}); | **Event:** *created*

var numeric = new ej.inputs.NumericTextBox({
value: 120,
created: function(){}
});
numeric.appendTo("#numeric"); |

| Adding custom classes | **property:** *cssClass*

\$("#numeric").ejNumericTextBox({
value: 100,
cssClass: "custom"
}); | **Property:** *cssClass*

var numeric = new ej.inputs.NumericTextBox({
value: 100,
cssClass: "custom"
});
numeric.appendTo("#numeric"); |

| Triggers when editor is destroyed | **Event:** *destroy*

\$("#numeric").ejNumericTextBox({
value: 120,
destroy: function(){}
}); | **Event:** *destroyed*

var numeric = new ej.inputs.NumericTextBox({
value: 120,
destroyed: function(){}
});
numeric.appendTo("#numeric"); |

| Destroys textbox | **Method:** *destroy*

\$("#numeric").ejNumericTextBox({
value: 100
});
var numericObj = \$("#numeric").data("ejNumericTextBox");
numericObj.destroy(); | **Method:** *destroy*

var numeric = new ej.inputs.NumericTextBox({
value: 100,
enabled: false
});
numeric.appendTo("#numeric");
numeric.destroy(); |

| Control state | **property:** *enabled*

\$("#numeric").ejNumericTextBox({
value: 100,
enabled: false
}); | **Property:** *enabled*

var numeric = new ej.inputs.NumericTextBox({
value: 100,
enabled: false
});
numeric.appendTo("#numeric"); |

| Persistence | **property:** *enablePersistence*

\$("#numeric").ejNumericTextBox({
value: 100,
enablePersistence: true
}); | **Property:** *enablePersistence*

var numeric = new ej.inputs.NumericTextBox({
value: 100,
enablePersistence: true
});
numeric.appendTo("#numeric"); |

| Right To Left | **property:** *enableRTL*

\$("#numeric").ejNumericTextBox({
value: 100,
enableRTL: true
}); | **Property:** *enableRtl*

var numeric = new ej.inputs.NumericTextBox({
value: 100,
enableRtl: true
});
numeric.appendTo("#numeric"); |

| Triggers when editor is focused in | **Event:** *focusIn*

\$("#numeric").ejNumericTextBox({
value: 20,
focusIn: function(){}
}); | **Event:** *focus*

var numeric = new ej.inputs.NumericTextBox({
value: 100,
focus: function(){}
});
numeric.appendTo("#numeric"); |

| Triggers when editor is focused out | **Event:** *focusOut*

\$("#numeric").ejNumericTextBox({
value: 100,
focusOut: function(){}
}); | **Event:** *blur*

var numeric = new ej.inputs.NumericTextBox({
value: 100,
blur: function(){}
});
numeric.appendTo("#numeric"); |

| Sets Height | **property:** *height*

\$("#numeric").ejNumericTextBox({
value: 100,
height: "40px"
}); | **Can be achieved using,**

var numeric = new ej.inputs.NumericTextBox({
value: 100,
cssClass: "custom"
});
numeric.appendTo("#numeric");
CSS
.e-numerictextbox.custom{
height: 40px;
}

| HTML Attributes | **property:** *htmlAttributes*

\$("#numeric").ejNumericTextBox({
value: 100,
htmlAttributes: {disabled: "disabled"}
}); | **Can be achieved using,**

<input id="numeric" type="text" name="textbox" />
var numeric = new ej.inputs.NumericTextBox({
value: 100
});
numeric.appendTo("#numeric") |

| Name of editor | **property:** *name*

\$("#numeric").ejNumericTextBox({
value: 100,
name: "Textbox"
}); | **Can be achieved using,**

HTML
<input id="numeric" type="text" name="textbox" />
var numeric = new ej.inputs.NumericTextBox({
value: 100,
});
numeric.appendTo("#numeric"); |

| Read only | **property:** *readOnly*

\$("#numeric").ejNumericTextBox({
value: 80,
readOnly: true
}); | **Property:** *readonly*

var numeric = new ej.inputs.NumericTextBox({
value: 80,
readonly: true
});
numeric.appendTo("#numeric"); |

| Rounded corners | **property:** *showRoundedCorner*

\$("#numeric").ejNumericTextBox({
value: 80,
showRoundedCorner: true
}); | **Can be achieved using**
var numeric = new ej.inputs.NumericTextBox({
value: 100,
cssClass: "e-style"
});
numeric.appendTo("#numeric");
CSS
.e-control-wrapper.e-numeric.e-input-group.e-style {
border: 2.5px solid;
border-radius: 1rem;
padding-left: 12px;
}
.e-control-wrapper.e-numeric.e-float-input.e-style .e-float-line::before, .e-control-wrapper.e-numeric.e-float-

input.e-style .e-float-line::after{
background: none ;

.e-control-wrapper.e-numeric.e-input-group.e-style.e-input-focus{
 border: solid grey !important;
} |

| Spin Button | **property:** *showSpinButton*

\$("#numeric").ejNumericTextBox({
value: 20,
showSpinButton: false
}); | **Property:** *showSpinButton*

var numeric = new ej.inputs.NumericTextBox({
value: 20,
showSpinButton: false
});
numeric.appendTo("#numeric"); |

| Width | **property:** *width*

\$("#numeric").ejNumericTextBox({
value: 20,
width: "220px"
}); | **Property:** *width*

var numeric = new ej.inputs.NumericTextBox({
value: 20,
width: "220px"
});
numeric.appendTo("#numeric"); |

| Clear Button | Not Applicable | **Property:** *showClearButton*

var numeric = new ej.inputs.NumericTextBox({
value: 100,
showClearButton: true
});
numeric.appendTo("#numeric"); |

Globalization

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Localization culture | **property:** *locale*

\$("#numeric").ejNumericTextBox({
value: 80,
locale: "de-DE"
}); | **Property:** *locale*

var numeric = new ej.inputs.NumericTextBox({
value: 80,
locale: "de-DE"
});
numeric.appendTo("#numeric"); |

Group

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Group digits in editor | **property:** *groupSize*

\$("#numeric").ejNumericTextBox({
value: 100,
groupSize: "2"
}); | Not Applicable |

| Group Separator | **property:** *groupSeparator*

\$("#numeric").ejNumericTextBox({
value: 100,
groupSeparator: "-"
}); | Not Applicable |

Numeric configuration

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Triggers on value change | **Event:** *change*

\$("#numeric").ejNumericTextBox({
value: 120,
change: function(){}
}); | **Event:** *change*

var numeric = new ej.inputs.NumericTextBox({
value: 120,
change: function(){}
});
numeric.appendTo("#numeric"); |

| Sets digits allowed after decimal point | **property:** *decimalPlaces*

\$("#numeric").ejNumericTextBox({
value: 100,
decimalPlaces: 2
}); | **Property:** *decimals*

var numeric = new ej.inputs.NumericTextBox({
value: 100,
format: "n2",
decimals: 2
});
numeric.appendTo("#numeric"); |

| Decrement value | Not Applicable | **Method:** *decrement*

var numeric = new ej.inputs.NumericTextBox({
value: 100
});
numeric.appendTo("#numeric");
numeric.decrement(); |

| Disable the textbox | **Method:** *disable*
 <code>\$("#numeric").ejNumericTextBox({value: 200});</code>
 <code>var numericObj = \$("#numeric").data("ejNumericTextBox");</code>
 <code>numericObj.disable();</code> | **Can be achieved using,** **API:** *enabled*
 <code>var numeric = new ej.inputs.NumericTextBox({value: 100});</code>
 <code>numeric.appendTo("#numeric");</code>
 <code>var numObj = document.getElementById("numeric").ej2_instances[0]</code>
 <code>numObj.enabled = false;</code> |

| Enable the textbox | **Method:** *enable*
 <code>\$("#numeric").ejNumericTextBox({value: 200});</code>
 <code>var numericObj = \$("#numeric").data("ejNumericTextBox");</code>
 <code>numericObj.enable();</code> | **Can be achieved using,** **API:** *enabled*
 <code>var numeric = new ej.inputs.NumericTextBox({value: 100});</code>
 <code>numeric.appendTo("#numeric");</code>
 <code>var numObj = document.getElementById("numeric").ej2_instances[0]</code>
 <code>numObj.enabled = true;</code> |

| Gets value of editor | **Method:** *getValue*
 <code>\$("#numeric").ejNumericTextBox({value: 100});</code>
 <code>var numericObj = \$("#numeric").data("ejNumericTextBox");</code>
 <code>numericObj.getValue();</code> | **Method:** *getText*
 <code>var numeric = new ej.inputs.NumericTextBox({value: 100});</code>
 <code>numeric.appendTo("#numeric");</code>
 <code>numeric.getText();</code> |

| Increment value | Not Applicable | **Method:** *increment*
 <code>var numeric = new ej.inputs.NumericTextBox({value: 80});</code>
 <code>numeric.appendTo("#numeric");</code>
 <code>numeric.increment();</code> |

| Step value | **property:** *incrementStep*
 <code>\$("#numeric").ejNumericTextBox({value: 100, incrementStep: 2});</code> | **Property:** *step*
 <code>var numeric = new ej.inputs.NumericTextBox({value: 100, step: 2});</code>
 <code>numeric.appendTo("#numeric");</code> |

| Sets Maximum value | **property:** *maxValue*
 <code>\$("#numeric").ejNumericTextBox({value: 100, maxValue: 200});</code> | **Property:** *max*
 <code>var numeric = new ej.inputs.NumericTextBox({value: 100, max: 200});</code>
 <code>numeric.appendTo("#numeric");</code> |

| Sets Minimum value | **property:** *minValue*
 <code>\$("#numeric").ejNumericTextBox({value: 100, minValue: 20});</code> | **Property:** *min*
 <code>var numeric = new ej.inputs.NumericTextBox({value: 100, min: 20});</code>
 <code>numeric.appendTo("#numeric");</code> |

| Negative pattern for formatting values | **property:** *negativePattern*
 <code>\$("#numeric").ejNumericTextBox({value: -20, negativePattern: "(n)"}</code> | Not Applicable |

| Positive pattern for formatting values | **property:** *positivePattern*
 <code>\$("#numeric").ejNumericTextBox({value: 20, positivePattern: "n kg"}</code> | Not Applicable |

| Specifies value | **property:** *value*
 <code>\$("#numeric").ejNumericTextBox({value: 100});</code> | **Property:** *value*
 <code>var numeric = new ej.inputs.NumericTextBox({value: 100});</code>
 <code>numeric.appendTo("#numeric");</code> |

| Displays hint on editor | **property:** *watermarkText*
 <code>\$("#numeric").ejNumericTextBox({value: 80, watermarkText: "Enter value:"});</code> | **Property:** *placeholder*
 <code>var numeric = new ej.inputs.NumericTextBox({value: 100, placeholder: "Enter value:"});</code>
 <code>numeric.appendTo("#numeric");</code> |

| Placeholder float type | Not Applicable | **Property:** *floatLabelType*
 ej.inputs.NumericTextBox({value: 200,placeholder: "Enter value:",floatLabelType: "Auto"});
 numeric.appendTo("#numeric"); |

Number Formats

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Set Currency symbol | **property:** *currencySymbol*
 ejCurrencyTextBox({value: 100,currencySymbol: "EUR"}); | **Property:** *currency*
 ej.inputs.NumericTextBox({value: 100,format: "c2",currency: "EUR"});
 currency.appendTo("#currency"); |

| Number Format | Not Applicable | **Property:** *format*
 ej.inputs.NumericTextBox({value: 200,format: "n2"});
 numeric.appendTo("#numeric"); |

Validation

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Strict Mode | **property:** *enableStrictMode*
 ejNumericTextBox({value: 80,enableStrictMode: true}); | **Property:** *strictMode*
 ej.inputs.NumericTextBox({value: 80,strictMode: true});
 numeric.appendTo("#numeric"); |

| Validation on typing | **property:** *validateOnType*
 ejNumericTextBox({value: 100,validateOnType: true}); | **Property:** *validateDecimalOnType*
 ej.inputs.NumericTextBox({value: 100,validateDecimalOnType: true});
 numeric.appendTo("#numeric"); |

| Validation Message | **property:** *validationMessage*
 ejNumericTextBox({value: 100,validationRules: {required: true},validationMessage: {required: "Required value"}}); | **Validation in NumericTextBox can be achieved through form validation**
 options = {rules: { 'numericRange': {required: [true, "Number is required"]}},
 formObject = new ej.inputs.FormValidator('#form-element', options);
 var customPlace= function(element, error) {
 element.parentNode.parentNode.appendChild(error);
 formObject.customPlacement = customPlace;
 var numeric = new ej.inputs.NumericTextBox({value: 15});
 numeric.appendTo('#numeric1');
HTML <form id="form-element" class="form-horizontal">
 <input id="numeric1" type="text" name="numericRange" class="form-control" />
 <div id="error"></div></form> |

| Validation Rules | **property:** *validationRules*
 ejNumericTextBox({value: 100,validationRules: {required: true}}); | **Validation in NumericTextBox can be achieved through form validation**
 options = {rules: { 'numericRange': {required: [true]}},
 formObject = new ej.inputs.FormValidator('#form-element', options);
 var customPlace= function(element, error) {
 element.parentNode.parentNode.appendChild(error);
 formObject.customPlacement = customPlace;
 var numeric = new ej.inputs.NumericTextBox({value: 15});
 numeric.appendTo('#numeric1');
HTML <form id="form-element"

```
class="form-horizontal"><br/><input id="numeric1" type="text" name="numericRange" class="form-control" /> <br/><div id="error"></div><br/></form> |
```

Pager

Getting started in EJ2 JavaScript Pager control

This section explains you the steps required to create a simple Essential JS 2 Pager and demonstrate the basic usage of the Pager control in a JavaScript application.

Dependencies

Below is the list of minimum dependencies required to use the Pager.

```
`javascript
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-grids
`
```

Setup for local environment

Refer the following steps for setup your local environment.

Step 1: Create a root folder `myapp` for your application.

Step 2: Create `myapp/resources` folder to store local scripts and styles files.

Step 3: Create `myapp/index.js` and `myapp/index.html` files for initializing Essential JS 2 Pager control.

Adding Syncfusion resources

The Essential JS 2 Pager control can be initialized by using either of the following ways.

- Using local script and style.
- Using CDN link for script and style.

Using local script and style

You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

After installing the Essential JS 2 product build, you can copy the pager and its dependencies scripts and style file into the `resources/scripts` and `resources/styles` folder.

Refer the below code to find location pager's script and style file.

Syntax:

Script: `(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js`

Styles: `(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css`

Example:

Script: C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-grids/dist/global/ej2-grids.min.js

Styles: C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-grids/styles/material.css

After copying the files, then you can refer the pager's scripts and styles into the `index.html` file.

```

`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Pager</title>
<!-- Essential JS 2 material base style -->
<link href="resources/styles/base/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 material theme -->
<link href="resources/styles/grid/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Pager's dependent global script -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<!-- Essential JS 2 Pager's global script -->
<script src="resources/scripts/ej2-grids.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`

```

Using CDN link for script and style

Using CDN link, you can directly refer the pager control's script and style into the `index.html`.

Refer the pager's CDN links as below

Syntax:

Script: `http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Styles: `http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

Example:

Script: <http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css>

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Pager</title>
<!-- Essential JS 2 base material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 pager material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Pager's dependent global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>
<!-- Essential JS 2 Pager's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js"
type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Adding Pager control

Now, you can start adding Pager control in the application. For getting started, add a `div` element for Pager control in `index.html`. Then refer the `index.js` file into the `index.html` file.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Pager</title>
<!-- Essential JS 2 base material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
```

```
<!-- Essential JS 2 pager material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Pager's dependent global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>
<!-- Essential JS 2 Pager's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js"
type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element for pager -->
<div id="Pager"></div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Place the following pager code in the `index.js`.

```
`javascript
var pager = ej.grids.Pager();
pager.appendTo('#Pager');
```

Page Size

`pageSize` value defines the number records to be displayed per page. The default value for the `pageSize` is 12.

INDEX.JS

```
var pager = new ej.grids.Pager({
    totalRecordsCount: 20,
    pageSize: 1
});
pager.appendTo('#Pager');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pager Component">
  <meta name="author" content="Syncfusion">
```

```

<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Pager"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Page Count

pageCount value defines the number of pages to be displayed in the pager component for navigation. The default value for **pageCount** is 10 and value will be updated based on **totalRecordsCount** and **pageSize** values.

INDEX.JS

```

var pager = new ej.grids.Pager({
    totalRecordsCount: 20,

```



```
    pageSize: 1,  
    pageCount: 3  
});  
pager.appendTo('#Pager');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 Pager</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="Typescript Pager Component">  
  <meta name="author" content="Syncfusion">  
  <link href="styles.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
base/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
grids/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
buttons/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
popups/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
navigations/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
dropdowns/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
lists/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
inputs/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
calendars/styles/material.css" rel="stylesheet">  
  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
splitbuttons/styles/material.css" rel="stylesheet">  
  
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"  
type="text/javascript"></script>  
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type  
="text/javascript"></script>  
</head>  
<body>  
  
  <div id="container">  
    <div id="Pager"></div>  
  </div>  
<script>  
var ele = document.getElementById('container');  
if(ele) {  
  ele.style.visibility = "visible";  
}
```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Run the application

Now, run the `index.html` in web browser, it will render the Essential JS 2 Pager control.

Output will be appears as follows.

INDEX.JS

```
var pager = new ej.grids.Pager({
  pageSize: 8,
  pageCount: 3,
  totalRecordsCount: 20,
});
pager.appendTo('#Pager');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pager</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pager Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Pager"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Style and appearance in EJ2 JavaScript Pager control

To modify the Pager appearance, you need to override the default CSS of Pager. Please find the CSS structure that can be used to modify the Pager appearance. Also, you have an option to create your own custom theme for all the JavaScript controls using our [Theme Studio](#).

Customizing the Pager

Use the below CSS to customize the Pager root element.

```

,

.e-pager {
background-color: #deecf9;
}
,

```

Customizing the Pager container element

Use the below CSS to customize the pager container element.

```

,

.e-pager .e-pagercontainer {
background-color: #deecf9;
}
,

```

Customizing the Pager navigation elements

Customize the pager navigation elements, using the below selector.

```

,

.e-pager .e-prevpagedisabled,
.e-pager .e-prevpage,
.e-pager .e-nextpage,

```

```
.e-pager .e-nextpagedisabled,  
.e-pager .e-lastpagedisabled,  
.e-pager .e-lastpage,  
.e-pager .e-firstpage,  
.e-pager .e-firstpagedisabled {  
background-color: #deecf9;  
}  
,
```

Customizing the Pager page numeric link elements

Use the below CSS to customize the pager current page numeric link elements.

```
,  
  
.e-pager .e-numericitem {  
border-radius: initial;  
}  
,
```

Customizing the Pager current page numeric element

Using this CSS, you can customize the pager current page numeric item.

```
,  
  
.e-pager .e-currentitem {  
background-color: #0078d7;  
}  
,
```

Accessibility in EJ2 JavaScript Pager control

Accessibility is achieved in the Pager component through WAI-ARIA standard and keyboard navigations. The Pager features can be effectively accessed through assistive technologies such as screen readers.

WAI-ARIA

WAI-ARIA (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components.

The following ARIA attributes are used in the Pager:

- pager (data-role)
- aria-selected (attribute)
- aria-owns (attribute)
- aria-label (attribute)

Keyboard navigation

Pager functionalities can be interactive with keyboard shortcuts.

The following keyboard shortcuts are supported by the Pager.

Interaction Keys | Description

Pager | |

Alt + J | Focus on the first pager item.

Tab / Shift + Tab | Focus on the next/previous pager item.

Enter / Space | Select the currently focused page.

Right Arrow / PageDown | Navigate to next page.

Left Arrow / PageUp | Navigate to previous page.

Home / End | Navigate to first and last page.

PDF Viewer

Getting Started

Getting started in Standalone PDF Viewer control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

Component Initialization with CDN link for script and style reference

Step 1: Create an app folder `myapp` for the Essential JS 2 JavaScript components.

Step 2: The Essential JS 2 component's global scripts and styles are already hosted in the below CDN link formats.

Syntax:

Script: `https://cdn.syncfusion.com/ej2/{Version}/dist/{PACKAGE_NAME}.min.js`

Styles: `https://cdn.syncfusion.com/ej2/{Version}/{PACKAGE_NAME}/styles/material.css`

Example:

Script: <https://cdn.syncfusion.com/ej2/23.1.36/dist/ej2.min.js>

Styles: <https://cdn.syncfusion.com/ej2/23.1.36/ej2-base/styles/material.css>

Note: While referring the scripts from the downloaded resources in your application, make sure to place the 'ej2-pdfviewer-lib' assets in the same directory as the 'ej2.min.js' script.

Step 3: Create a HTML page (index.html) in `myapp` location and add the CDN link references. Now, add the Div element and initiate the Essential JS 2 PDF Viewer component in the index.html by using following code.

INDEX.HTML

```

<!DOCTYPE html><html xmlns="http://www.w3.org/1999/xhtml"><head>
  <title>Essential JS 2</title>
  <!-- Essential JS 2 material theme -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pdfviewer/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <!-- Essential JS 2 PDF Viewer's script -->
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
  <body>
    <!--element which is going to render-->
    <div id="container">
      <div id="PdfViewer" style="height:580px;width:100%;"></div>
    </div>
    <script>
      //Initialize PDF Viewer component
      var pdfviewer = new ej.pdfviewer.PdfViewer({
documentPath:'https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf',
resourceUrl:'https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib'
      });
      ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar,ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);

      //PDF Viewer control rendering starts
      pdfviewer.appendTo('#PdfViewer');
    </script>
    <script>
      var ele = document.getElementById('container');
      if(ele) {
        ele.style.visibility = "visible";
      }
    </script>
  </script>

```

```
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
</body></html>
```

Step 4: Now, run the `index.html` in web browser, it will render the **Essential JS 2 PDF Viewer** component.

Limitation over Server-Backed PDF Viewer to Standalone PDF Viewer control

When comparing a Standalone PDF Viewer to a Server-Backed PDF Viewer control, it's crucial to understand the limitations that the Standalone PDF Viewer may have in comparison. These limitations are important to consider

PNG Image Support

The Standalone PDF Viewer does not have the capability to utilize PNG format for adding images to handwritten annotations ,custom stamp ,signature and initial form fields. It's important to be aware that only certain image formats, such as JPEG, are compatible for these purposes.

Local File Access

- The Standalone PDF Viewer control does not have the capability to directly access and load local physical files from a user's device. As a result, it is not possible to use a `documentPath` to load a PDF file directly from a local server within the viewer.
- The Standalone PDF Viewer allows users to export annotations and form fields from the viewer, it's important to be aware that the viewer does not support the direct import of annotations and form fields from a locally specified file path. In other words, you can extract annotations and form fields from the viewer, but you cannot reintroduce them into the viewer from external sources by specifying a file path located on your local device.

Note: These limitations are temporary and are expected to be addressed in the near future.

Getting started in EJ2 JavaScript PDF Viewer control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

Component Initialization with CDN link for script and style reference

Step 1: Create an app folder `myapp` for the Essential JS 2 JavaScript components.

Step 2: The Essential JS 2 component's global scripts and styles are already hosted in the below CDN link formats.

Syntax:

Script: `https://cdn.syncfusion.com/ej2/{Version}/dist/{PACKAGE_NAME}.min.js`

Styles: `https://cdn.syncfusion.com/ej2/{Version}/{PACKAGE_NAME}/styles/material.css`

Example:

Script: <https://cdn.syncfusion.com/ej2/23.1.36/dist/ej2.min.js>

Styles: <https://cdn.syncfusion.com/ej2/23.1.36/ej2-base/styles/material.css>

Step 3: Create a HTML page (index.html) in myapp location and add the CDN link references. Now, add the Div element and initiate the Essential JS 2 PDF Viewer component in the index.html by using following code.

INDEX.HTML

```
<!DOCTYPE html><html xmlns="http://www.w3.org/1999/xhtml"><head>
  <title>Essential JS 2</title>
  <!-- Essential JS 2 material theme -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pdfviewer/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <!-- Essential JS 2 PDF Viewer's script -->
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
  <body>
    <!--element which is going to render-->
    <div id="container">
      <div id="PdfViewer" style="height:580px;width:100%;">
      </div>
    </div>
    <script>
      //Initialize PDF Viewer component
      var pdfviewer = new ej.pdfviewer.PdfViewer({
        documentPath:
"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
        serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
      });
      ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar,
```



```

ej.pdfviewer.Magnification,
ej.pdfviewer.Annotation, ej.pdfviewer.FormDesigner,
ej.pdfviewer.FormFields);
    //PDF Viewer control rendering starts
    pdfviewer.appendTo('#PdfViewer');
</script>
<script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
</body></html>

```

Note: We have provided the support to dynamically change the `serviceURL`. So, after changing the `serviceURL` dynamically, you need invoke the `pdfViewer.dataBind()` method to update the `serviceURL` quickly. This will effectively change the `serviceURL` dynamically. Ensure that this step is performed after version 23.1.36.

```

document.getElementById('load').addEventListener('click', function () {
pdfViewer.serviceUrl = "https://services.syncfusion.com/angular/production/api/pdfviewer";
pdfViewer.documentPath = "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf";
pdfViewer.dataBind();
pdfViewer.load(pdfViewer.documentPath, null);
});

```

Step 4: Now, run the `index.html` in web browser, it will render the `Essential JS 2 PDF Viewer` component.

For PDF Viewer `serviceUrl` creation, follow the steps provided in the [link](#)

How to run the PDF Viewer web service

1. Download the sample from the [Web service sample in GitHub](#) link.
2. Navigate to the `ASP.NET Core` folder and open it in the command prompt.
3. Use the below command to restore the required packages.

```
dotnet restore
```

4. Use the below command to run the web service.

dotnet run

5. You can see that the PDF Viewer server instance runs in the localhost with the port number localhost:5001 and navigate to the PDF Viewer Web control localhost:5001/pdfviewer which returns the default get response method. We can bind the link to the `serviceUrl` property of PDF Viewer as below.

```
`javascript
var pdfviewer = new ej.pdfviewer.PdfViewer({
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: 'https://localhost:5001/pdfviewer'
});
```

Note: When configuring the server-backed PDF viewer, it's essential to understand that there is no need to include the pdfium.js and pdfium.wasm files. Unlike the standalone PDF viewer, which relies on these files for local rendering, the server-backed PDF viewer fetches and renders PDFs directly from the server. Consequently, you can exclude the copy command for deployment process, as they are not required to load and display PDFs in this context.

Feature module in EJ2 JavaScript Pdfviewer control

The PDF Viewer features are segregated into individual feature-wise modules to enable selectively referencing in the application. The required modules should be injected to extend its functionality. The following are the selective modules of PDF Viewer that can be included as required:

The available PdfViewer modules are:

- **Toolbar:**- Built-in toolbar for better user interaction.
- **Magnification:**- Perform zooming operation for better viewing experience.
- **Navigation:**- Easy navigation across the PDF pages.
- **LinkAnnotation:**- Easy navigation within and outside of the PDF document.
- **ThumbnailView:**- Easy navigation with in the PDF document.
- **BookmarkView:**- Easy navigation based on the bookmark content of the PDF document.
- **TextSelection:**- Select and copy text from a PDF file.
- **TextSearch:**- Search a text easily across the PDF document.
- **Print:**- Print the entire document or a specific page directly from the browser.
- **Annotation:**- Annotations can be added or edited in the PDF document.
- **FormFields:**- Preserve the form fields in the PDF document.
- **FormDesigner:**- Form fields can be added or edited in the PDF document.

In addition to injecting the required modules in your application, enable corresponding properties to extend the functionality for a PDF Viewer instance.

Refer to the following table.

| Module | Dependent modules to be injected for extending the functionality of PDF Viewer in your application | Property to enable the functionality for a PDF Viewer instance |

|---|---|---|

```
|Toolbar|PdfViewer.Inject(Toolbar)|let pdfViewer: PdfViewer = new PdfViewer({ enableToolbar: true });|
```

```
|Magnification|PdfViewer.Inject(Magnification)|let pdfViewer: PdfViewer = new PdfViewer({ enableMagnification: true });|
```

```
|Navigation|PdfViewer.Inject(Navigation)|let pdfViewer: PdfViewer = new PdfViewer({ enableNavigation: true });|
```

```
|LinkAnnotation|PdfViewer.Inject(LinkAnnotation)|let pdfViewer: PdfViewer = new PdfViewer({ enableHyperlink: true });|
```

```
|ThumbnailView|PdfViewer.Inject(ThumbnailView)|let pdfViewer: PdfViewer = new PdfViewer({ enableThumbnail: true });|
```

```
|BookmarkView|PdfViewer.Inject(BookmarkView)|let pdfViewer: PdfViewer = new PdfViewer({ enableBookmark: true });|
```

```
|TextSelection|PdfViewer.Inject(TextSelection)|let pdfViewer: PdfViewer = new PdfViewer({ enableTextSelection: true });|
```

```
|TextSearch|PdfViewer.Inject(TextSearch)|let pdfViewer: PdfViewer = new PdfViewer({ enableTextSearch: true });|
```

```
|Print|PdfViewer.Inject(Print)|let pdfViewer: PdfViewer = new PdfViewer({ enablePrint: true });|
```

```
|Annotation|PdfViewer.Inject(Annotation)|let pdfViewer: PdfViewer = new PdfViewer({ enableAnnotation: true });|
```

```
|FormFields|PdfViewer.Inject(FormFields)|let pdfViewer: PdfViewer = new PdfViewer({ enableFormFields: true });|
```

```
|FormDesigner|PdfViewer.Inject(FormDesigner)|let pdfViewer: PdfViewer = new PdfViewer({ enableFormDesigner: true });|
```

See also

- [Toolbar items](#)
- [Toolbar customization](#)

Open PDF files

You might need to open and view the PDF files from various location. In this section, you can find the information about how to open PDF files from URL, database and as base64 string.

Opening a PDF from URL

If you have your PDF files in the web, you can open it in the viewer using URL.

Step 1: Create a Simple PDF Viewer Sample in JavaScript

Start by following the steps provided in this [link](#) to create a simple PDF viewer sample in JavaScript. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the PdfViewerController.cs File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this [link](#) for instructions on how to create a web service project.
2. Open the PdfViewerController.cs file in your web service project.
3. Modify the Load() method to open it in the viewer using URL

```
`csharp
public IActionResult Load([FromBody] Dictionary<string, string> jsonData)
{
    // Initialize the PDF viewer object with memory cache object
    PdfRenderer pdfviewer = new PdfRenderer(_cache);
    MemoryStream stream = new MemoryStream();
    object jsonResult = new object();
    if (jsonObject != null && jsonObject.ContainsKey("document"))
    {
        if (bool.Parse(jsonObject["isFileName"]))
        {
            string documentPath = GetDocumentPath(jsonData["document"]);
            if (!string.IsNullOrEmpty(documentPath))
            {
                byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
                stream = new MemoryStream(bytes);
            }
            else
            {
                string fileName = jsonData["document"].Split(new string[] { "://" }, StringSplitOptions.None)[0];
                if (fileName == "http" || fileName == "https")
                {
                    WebClient WebClient = new WebClient();
                    byte[] pdfDoc = WebClient.DownloadData(jsonData["document"]);
                    stream = new MemoryStream(pdfDoc);
                }
            }
        }
    }
}
```

```

else
{
return this.Content(jsonData["document"] + " is not found");
}
}
}
else
{
byte[] bytes = Convert.FromBase64String(jsonObject["document"]);
stream = new MemoryStream(bytes);
}
}
jsonResult = pdfviewer.Load(stream, jsonObject);
return Content(JsonConvert.SerializeObject(jsonResult));
}
`

```

Step 3: Set the PDF Viewer Properties in React PDF viewer component

Modify the `serviceUrl` property of the PDF viewer component with the accurate URL of your web service project, replacing `https://localhost:44396/pdfviewer` with the actual URL of your server. Modify the `documentPath` with the correct PDF Document URL want to load.

```

`javascript
import { PdfViewer, Toolbar, Magnification, Navigation, LinkAnnotation, ThumbnailView,
BookmarkView, TextSelection, Annotation, FormFields, FormDesigner} from '@syncfusion/ej2-
pdfviewer';

PdfViewer.Inject( Toolbar, Magnification, Navigation, LinkAnnotation, ThumbnailView,
BookmarkView, TextSelection, Annotation, FormFields, FormDesigner);

let viewer: PdfViewer = new PdfViewer();

// Replace correct PDF Document URL want to load
viewer.documentPath="https://cdn.syncfusion.com/content/PDFViewer/flutter-succinctly.pdf"

// Replace the "localhost:44396" with the actual URL of your server
viewer.serviceUrl = 'https://localhost:44396/pdfviewer';
viewer.appendTo('#pdfViewer');
`

```

[View sample in GitHub](#)

Opening a PDF from base64 data

The following steps explain how the PDF file can be loaded in PDF Viewer as base64 string.

Step 1: Create a Simple PDF Viewer Sample in Angular

Start by following the steps provided in this [link](#) to create a simple PDF viewer sample in Angular. This will give you a basic setup of the PDF viewer component.

Step 2: Use the following code snippet to load PDF document using base64 string.

```
,
<button id='load'>LoadDocumentFromBase64</button>
,
`javascript
// Load PDF document from Base64 string
document.getElementById('load').addEventListener('click', () => {
viewer.load(
'data:application/pdf;base64,'+ AddBase64String, null);
}
,
```

[View sample in GitHub](#)

Opening a PDF from database

To load a PDF file from SQL Server database in a PDF Viewer, you can follow the steps below

Step 1: Create a Simple PDF Viewer Sample in JavaScript

Start by following the steps provided in this [link](#) to create a simple PDF viewer sample in JavaScript. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the PdfViewerController.cs File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this [link](#) for instructions on how to create a web service project.
2. Open the PdfViewerController.cs file in your web service project.
3. Import the required namespaces at the top of the file:

```
`csharp
using System.IO;
using System.Data.SqlClient;
,
```

4. Add the following private fields and constructor parameters to the PdfViewerController class, In the constructor, assign the values from the configuration to the corresponding fields

```
`csharp
```

```

private IConfiguration _configuration;
public readonly string _connectionString;
public PdfViewerController(IWebHostEnvironment hostingEnvironment, IMemoryCache cache,
IConfiguration configuration)
{
    _hostingEnvironment = hostingEnvironment;
    _cache = cache;
    _configuration = configuration;
    connectionString = configuration.GetValue<string>("ConnectionString");
}
`

```

5. Modify the `Load()` method to open it in the viewer using URL

```

`csharp
public IActionResult Load([FromBody] Dictionary<string, string> jsonData)
{
    // Initialize the PDF viewer object with memory cache object
    PdfRenderer pdfviewer = new PdfRenderer(_cache);
    MemoryStream stream = new MemoryStream();
    object jsonResult = new object();
    if (jsonObject != null && jsonObject.ContainsKey("document"))
    {
        if (bool.Parse(jsonObject["isFileName"]))
        {
            string documentPath = GetDocumentPath(jsonData["document"]);
            if (!string.IsNullOrEmpty(documentPath))
            {
                byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
                stream = new MemoryStream(bytes);
            }
            string documentName = jsonObject["document"];
            string connectionString = _connectionString;
            System.Data.SqlClient.SqlConnection connection = new
            System.Data.SqlClient.SqlConnection(connectionString);

```

```
//Searches for the PDF document from the database
string query = "SELECT FileData FROM Table WHERE FileName = '" + documentName + "'";
System.Data.SqlClient.SqlCommand command = new System.Data.SqlClient.SqlCommand(query,
connection);
connection.Open();
using (SqlDataReader reader = command.ExecuteReader())
{
    if (reader.Read())
    {
        byte[] byteArray = (byte[])reader["FileData"];
        stream = new MemoryStream(byteArray);
    }
}
else
{
    byte[] bytes = Convert.FromBase64String(jsonObject["document"]);
    stream = new MemoryStream(bytes);
}
jsonResult = pdfviewer.Load(stream, jsonObject);
return Content(JsonConvert.SerializeObject(jsonResult));
}
```

6. Open the `appsettings.json` file in your web service project, Add the following lines below the existing `"AllowedHosts"` configuration

```
`json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  }
}
```



```

},
"AllowedHosts": "*",
"ConnectionString": "Your connection string for SQL server"
}
`

```

Note: Replace **Your Connection string from SQL server** with the actual connection string for your SQL Server database

Note: The **System.Data.SqlClient** package must be installed in your application to use the previous code example. You need to modify the connectionString variable in the previous code example as per the connection string of your database.

[View sample in GitHub](#)

Saving PDF file

After editing the PDF file with various annotation tools, you will need to save the updated PDF to the server, database, or local file system.

Save PDF file to Server

Need to save the modified PDF back to a server. To achieve this, proceed with the following steps

Step 1: Create a Simple PDF Viewer Sample in JavaScript

Start by following the steps provided in this [link](#) to create a simple PDF viewer sample in JavaScript. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the PdfViewerController.cs File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this [link](#) for instructions on how to create a web service project.
2. Open the PdfViewerController.cs file in your web service project.
3. Modify the Download() method to open it in the viewer using URL

```

`csharp
public IActionResult Download([FromBody] Dictionary<string, string> jsonObject)
{
    //Initialize the PDF Viewer object with memory cache object
    PdfRenderer pdfviewer = new PdfRenderer(_cache);
    string documentBase = pdfviewer.GetDocumentAsBase64(jsonObject);
    MemoryStream stream = new MemoryStream();
    string documentName = jsonObject["document"];
    string result = Path.GetFileNameWithoutExtension(documentName);
    string fileName = result + "_downloaded.pdf";
    // Save the file on the server

```

```

string serverFilePath = @"Path to where you need to save your file in the server";
string filePath = Path.Combine(serverFilePath, fileName);
using (FileStream fileStream = new FileStream(filePath, FileMode.Create))
{
    //Saving the new file in root path of application
    stream.CopyTo(fileStream);
    fileStream.Close();
}
return Content(documentBase);
}
`

```

Step 3: Set the PDF Viewer Properties in React PDF viewer component

Modify the `serviceUrl` property of the PDF viewer component with the accurate URL of your web service project, replacing `https://localhost:44396/pdfviewer` with the actual URL of your server. Modify the `documentPath` with the correct PDF Document URL want to load.

```

`javascript
import { PdfViewer, Toolbar, Magnification, Navigation, LinkAnnotation, ThumbnailView,
BookmarkView, TextSelection, Annotation, FormFields, FormDesigner } from '@syncfusion/ej2-
pdfviewer';

PdfViewer.Inject( Toolbar, Magnification, Navigation, LinkAnnotation, ThumbnailView,
BookmarkView, TextSelection, Annotation, FormFields, FormDesigner);

let viewer: PdfViewer = new PdfViewer();

// Replace PDF_Succinctly.pdf with the actual document name that you want to load
viewer.documentPath = "PDF_Succinctly.pdf"
viewer.serviceUrl = 'https://localhost:44396/pdfviewer';
viewer.appendTo('#pdfViewer');
`

```

[View sample in GitHub](#)

Download PDF file as a copy

In the built-in toolbar, you have an option to download the updated PDF to the local file system, you can use it to download the PDF file.

```

`html
<button id="download">Download</button>
`

`javascript

```

```
document.getElementById('download').addEventListener('click', function () {
//API to perform download action.
viewer.download();
});
`
```

Save PDF file to Database

If you have plenty of PDF files stored in database and you want to save the updated PDF file back to the database, use the following code example.

Step 1: Create a Simple PDF Viewer Sample in JavaScript

Start by following the steps provided in this [link](#) to create a simple PDF viewer sample in JavaScript. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the PdfViewerController.cs File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this [link](#) for instructions on how to create a web service project.
2. Open the PdfViewerController.cs file in your web service project.
3. Import the required namespaces at the top of the file:

```
`csharp
using System.IO;
using System.Data.SqlClient;
`
```

4. Add the following private fields and constructor parameters to the PdfViewerController class, In the constructor, assign the values from the configuration to the corresponding fields

```
`csharp
private IConfiguration _configuration;
public readonly string _connectionString;
public PdfViewerController(IWebHostEnvironment hostingEnvironment, IMemoryCache cache,
IConfiguration configuration)
{
    _hostingEnvironment = hostingEnvironment;
    _cache = cache;
    _configuration = configuration;
    connectionString = configuration.GetValue<string>("ConnectionString");
}
`
```

5. Modify the `Download()` method to open it in the viewer using URL

```
`csharp
[HttpPost("Download")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/Download")]
//Post action for downloading the PDF documents
public async Task<ActionResult> Download([FromBody] Dictionary<string, string> jsonObject)
{
    //Initialize the PDF Viewer object with memory cache object
    PdfRenderer pdfviewer = new PdfRenderer(_cache);
    string documentBase = pdfviewer.GetDocumentAsBase64(jsonObject);
    byte[] documentBytes = Convert.FromBase64String(documentBase.Split(",")[1]);
    string documentId = jsonObject["documentId"];
    string result = Path.GetFileNameWithoutExtension(documentId);
    string fileName = result + "_downloaded.pdf";
    string connectionString = _connectionString;
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        using (SqlCommand cmd = new SqlCommand("INSERT INTO Table (FileName, fileData) VALUES (@FileName, @fileData)", connection))
        {
            cmd.Parameters.AddWithValue("@FileName", fileName);
            cmd.Parameters.AddWithValue("@fileData", documentBytes);
            cmd.ExecuteNonQuery();
        }
        connection.Close();
    }
    return Content(documentBase);
}
```

6. Open the `appsettings.json` file in your web service project, Add the following lines below the existing `"AllowedHosts"` configuration

```

`json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionString": "Your connection string for SQL server"
}
`

```

Note: Replace **Your Connection string from SQL server** with the actual connection string for your SQL Server database

Note: The **System.Data.SqlClient** package must be installed in your application to use the previous code example. You need to modify the connectionString variable in the previous code example as per the connection string of your database.

[View sample in GitHub](#)

Toolbar in EJ2 JavaScript Pdfviewer control

The PDF Viewer comes with a powerful built-in toolbar to execute important actions such as page navigation, text search, view mode, download, print, bookmark, and thumbnails.

The following table shows built-in toolbar items and its actions:-

Option	Description
OpenOption	This option provides an action to load the PDF documents to the PDF Viewer.
PageNavigationTool	This option provides an action to navigate the pages in the PDF Viewer. It contains GoToFirstPage, GoToLastPage, GotoPage, GoToNext, and GoToLast tools.
MagnificationTool	This option provides an action to magnify the pages either with predefined or user defined zoom factors in the PDF Viewer. Contains ZoomIn, ZoomOut, Zoom, FitPage and FitWidth tools.
PanTool	This option provides an action for panning the pages in the PDF Viewer.
SelectionTool	This option provides an action to enable/disable the text selection in the PDF Viewer.
SearchOption	This option provides an action to search a word in the PDF documents.
PrintOption	This option provides an action to print the PDF document being loaded in the PDF Viewer.
DownloadOption	This Download option provides an action to download the PDF document that has been loaded in the PDF Viewer.

| UndoRedoTool | This tool provides options to undo and redo the annotation actions performed in the PDF Viewer. |

| AnnotationEditTool | This tool provides options to enable or disable the edit mode of annotation in the PDF Viewer. |

| FormDesignerEditTool | This tool provides options to enable or disable the edit mode of form fields in the PDF Viewer. |

Show/Hide the default toolbar

The PDF Viewer has an option to show or hide the complete default toolbar. You can achieve this by using following two ways.,

```
`html
{% raw %}
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 fabric theme -->
<link href="{{CDN_LINK}}ej2-pdfviewer/styles/fabric.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 PDF Viewer's global script -->
<script src="{{CDN_LINK}}dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!--element which is going to render-->
<div id='container'>
<div id='PdfViewer' style="height:500px;width:100%;">
</div>
</div>
</body>
</html>
{% endraw %}
`
```

- **Show/Hide toolbar using enableToolbar API as in the following code snippet**

INDEX.HTML

```
<!DOCTYPE html><html xmlns="http://www.w3.org/1999/xhtml"><head>
<title>Essential JS 2</title>
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pdfviewer/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<!-- Essential JS 2 PDF Viewer's global script -->
<script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<!--element which is going to render-->
<div id="container">
<div id="PdfViewer" style="height:500px;width:100%;">
</div>
</div>
<script>
// Initialize PDF Viewer component.
var pdfviewer = new ej.pdfviewer.PdfViewer({
enableToolbar: false,
documentPath:
"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
resourceUrl:'https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.Annotation, ej.pdfviewer.FormDesigner,
ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
</script>

<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
</body></html>

```

Note: To set up the **server-backed PDF Viewer**,

Add the below `serviceUrl` in the `index.html` file

`serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'`

- **Show/Hide toolbar using showToolbar as in the following code snippet**

INDEX.HTML

```
<!DOCTYPE html><html xmlns="http://www.w3.org/1999/xhtml"><head>
  <title>Essential JS 2</title>
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pdfviewer/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <!-- Essential JS 2 PDF Viewer's global script -->
  <script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
  <body>
    <button id="set">showToolbarItem</button>
    <!--element which is going to render-->
    <div id="container">
      <div id="PdfViewer" style="height:500px;width:100%;"></div>
    </div>
    <script>
      // Initialize PDF Viewer component.
      var pdfviewer = new ej.pdfviewer.PdfViewer({
        documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf",
        resourceUrl: 'https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-
pdfviewer-lib'
      });
      ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar,
```



```

ej.pdfviewer.Magnification,
ej.pdfviewer.Annotation, ej.pdfviewer.FormDesigner,
ej.pdfviewer.FormFields);
    pdfviewer.appendTo('#PdfViewer');
    document.getElementById('set').addEventListener('click', ()=> {
        pdfviewer.toolbar.showToolbar(false);
    });
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
</body></html>

```

Note: To set up the **server-backed PDF Viewer**,

Add the below `serviceUrl` in the `index.html` file

`serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'`

Show/Hide the default toolbaritem

The PDF Viewer has an option to show or hide these grouped items in the default toolbar.

- **Show/Hide toolbaritem using toolbarSettings as in the following code snippet.**

INDEX.HTML

```

<!DOCTYPE html><html xmlns="http://www.w3.org/1999/xhtml"><head>
    <title>Essential JS 2</title>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pdfviewer/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <!-- Essential JS 2 PDF Viewer's global script -->
    <script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>
  <!--element which is going to render-->
  <div id="container">
    <div id="PdfViewer" style="height:500px;width:100%;">
      </div>
    </div>
    <script>
      // Initialize PDF Viewer component.
      var pdfviewer = new ej.pdfviewer.PdfViewer({
        documentPath:
"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
resourceUrl:'https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib',
        toolbarSettings: { showTooltip : true, toolbarItems:
['OpenOption']}
      });
      ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.Annotation, ej.pdfviewer.FormDesigner,
ej.pdfviewer.FormFields);
      pdfviewer.appendTo('#PdfViewer');
    </script>

    <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
</body></html>

```

Note: To set up the **server-backed PDF Viewer**,

Add the below **serviceUrl** in the **index.html** file

serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'

- **Show/Hide toolbaritem using showToolbaritem as in the following code snippet**

INDEX.HTML

```

<!DOCTYPE html><html xmlns="http://www.w3.org/1999/xhtml"><head>
  <title>Essential JS 2</title>
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pdfviewer/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<!-- Essential JS 2 PDF Viewer's global script -->
<script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <button id="set">showToolbarItem</button>
  <!--element which is going to render-->
  <div id="container">
    <div id="PdfViewer" style="height:500px;width:100%;">
    </div>
  </div>
  <script>
    // Initialize PDF Viewer component.
    var pdfviewer = new ej.pdfviewer.PdfViewer({
      documentPath:
        "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
      resourceUrl: 'https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib'
    });
    ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
    ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
    ej.pdfviewer.Toolbar,
    ej.pdfviewer.Magnification,
    ej.pdfviewer.Annotation, ej.pdfviewer.FormDesigner,
    ej.pdfviewer.FormFields);
    pdfviewer.appendTo('#PdfViewer');
    document.getElementById('set').addEventListener('click',
    ()=> {
      pdfviewer.toolbar.showToolbarItem(["OpenOption"], false);
    });
  </script>

  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
</body></html>

```

Note: To set up the **server-backed PDF Viewer**,

Add the below `serviceUrl` in the `index.html` file

`serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'`

See also

- [Toolbar customization](#)
- [Feature Modules](#)

Navigation in EJ2 JavaScript Pdfviewer control

The ASP.NET Core PDF Viewer supports different internal and external navigations.

Toolbar page navigation option

The default toolbar of PDF Viewer contains the following navigation options

- [Go to page](#):- Navigates to the specific page of a PDF document.
- [Show next page](#):- Navigates to the next page of PDF a document.
- [Show previous page](#):- Navigates to the previous page of a PDF document.
- [Show first page](#):- Navigates to the first page of a PDF document.
- [Show last page](#):- Navigates to the last page of a PDF document.

```
`html
{% raw %}
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 fabric theme -->
<link href="{{CDN_LINK}}ej2-pdfviewer/styles/fabric.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 PDF Viewer's global script -->
<script src="{{CDN_LINK}}dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!--element which is going to render-->
<div id='container'>
<div id='PdfViewer' style="height:500px;width:100%;"></div>
</div>
</body>
</html>
{% endraw %}
`
```

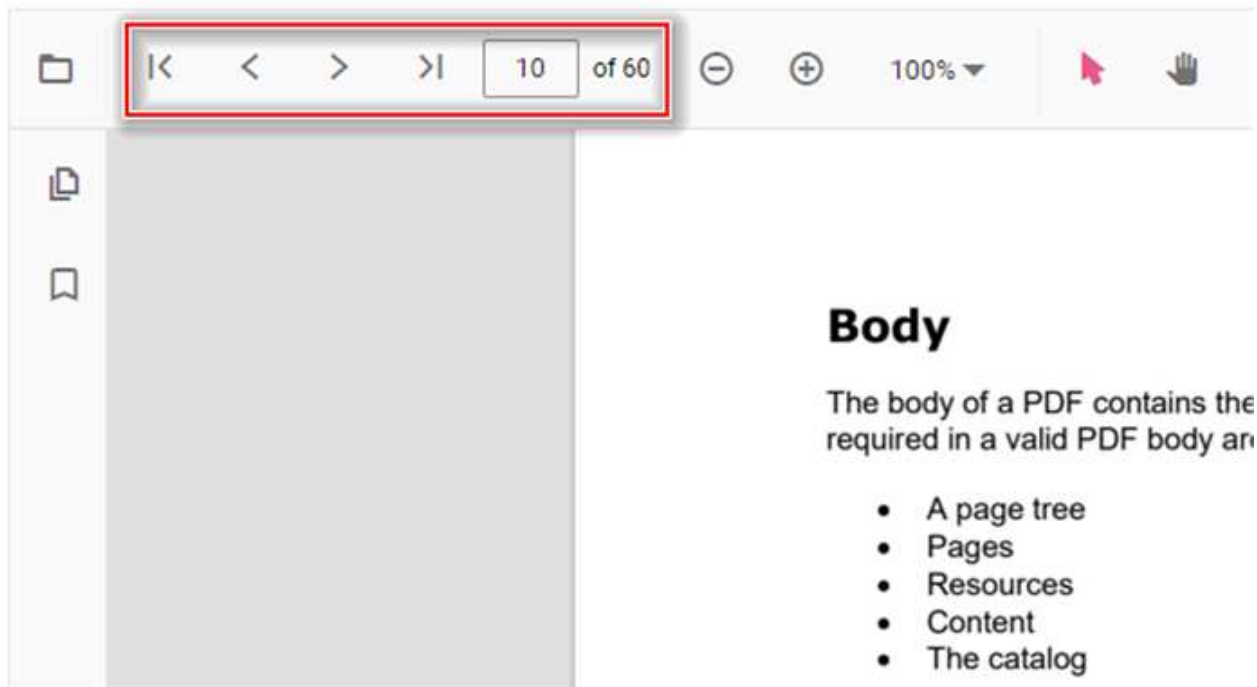
You can enable/disable page navigation option in PDF Viewer using the following code snippet.,

STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableNavigation: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableNavigation: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```



Also, you can programmatically perform page navigation options as follows.

```
`html
{% raw %}
```

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 fabric theme -->
<link href="{{:CDN_LINK}}ej2-pdfviewer/styles/fabric.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 PDF Viewer's global script -->
<script src="{{:CDN_LINK}}dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!--element which is going to render-->
<div id='container'>
<button id="goToFirstPage">Go To First Page</button>
<button id="goToLastPage">Go To last Page</button>
<button id="goToNextPage">Go To Next Page</button>
<button id="goToPage">Go To Page</button>
<button id="goToPreviousPage">Go To Previous Page</button>
<div id='PdfViewer' style="height:500px;width:100%;"></div>
</div>
</body>
</html>
{% endraw %}
`

```

STANDALONE

```

var viewer = new ej.pdfviewer.PdfViewer({
  documentPath: 'https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf',
});
ej.pdfviewer.PdfViewer.Inject(
  ej.pdfviewer.Toolbar,
  ej.pdfviewer.Magnification,
  ej.pdfviewer.BookmarkView,
  ej.pdfviewer.ThumbnailView,
  ej.pdfviewer.TextSelection,
  ej.pdfviewer.TextSearch,
  ej.pdfviewer.Print,
  ej.pdfviewer.Navigation,
  ej.pdfviewer.LinkAnnotation,
  ej.pdfviewer.Annotation,
  ej.pdfviewer.FormFields

```

```
);
viewer.appendTo('#pdfViewer');
// Go To First Page
document.getElementById('goToFirstPage').addEventListener('click', () => {
viewer.navigation.goToFirstPage();
});
// Go To Last Page
document.getElementById('goToLastPage').addEventListener('click', () => {
viewer.navigation.goToLastPage();
});
// Go To Next Page
document.getElementById('goToNextPage').addEventListener('click', () => {
viewer.navigation.goToNextPage();
});
// Go To Page
document.getElementById('goToPage').addEventListener('click', () => {
viewer.navigation.goToPage(4);
});
// Go To Previous Page
document.getElementById('goToPreviousPage').addEventListener('click', () => {
{
viewer.navigation.goToPreviousPage();
});
});
```

SERVER-BACKED

```
var viewer = new ej.pdfviewer.PdfViewer({
documentPath: 'https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf',
serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(
ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.BookmarkView,
ej.pdfviewer.ThumbnailView,
ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch,
ej.pdfviewer.Print,
ej.pdfviewer.Navigation,
ej.pdfviewer.LinkAnnotation,
ej.pdfviewer.Annotation,
ej.pdfviewer.FormFields
);
viewer.appendTo('#pdfViewer');
// Go To First Page
document.getElementById('goToFirstPage').addEventListener('click', () => {
viewer.navigation.goToFirstPage();
});
// Go To Last Page
document.getElementById('goToLastPage').addEventListener('click', () => {
viewer.navigation.goToLastPage();
});
// Go To Next Page
document.getElementById('goToNextPage').addEventListener('click', () => {
viewer.navigation.goToNextPage();
});
});
```

```
// Go To Page
document.getElementById('goToPage').addEventListener('click', () => {
viewer.navigation.goToPage(4);
});
// Go To Previous Page
document.getElementById('goToPreviousPage').addEventListener('click', () => {
{
viewer.navigation.goToPreviousPage();
}});
```

Find the [here](#) to perform the page navigation options programmatically.

Bookmark navigation

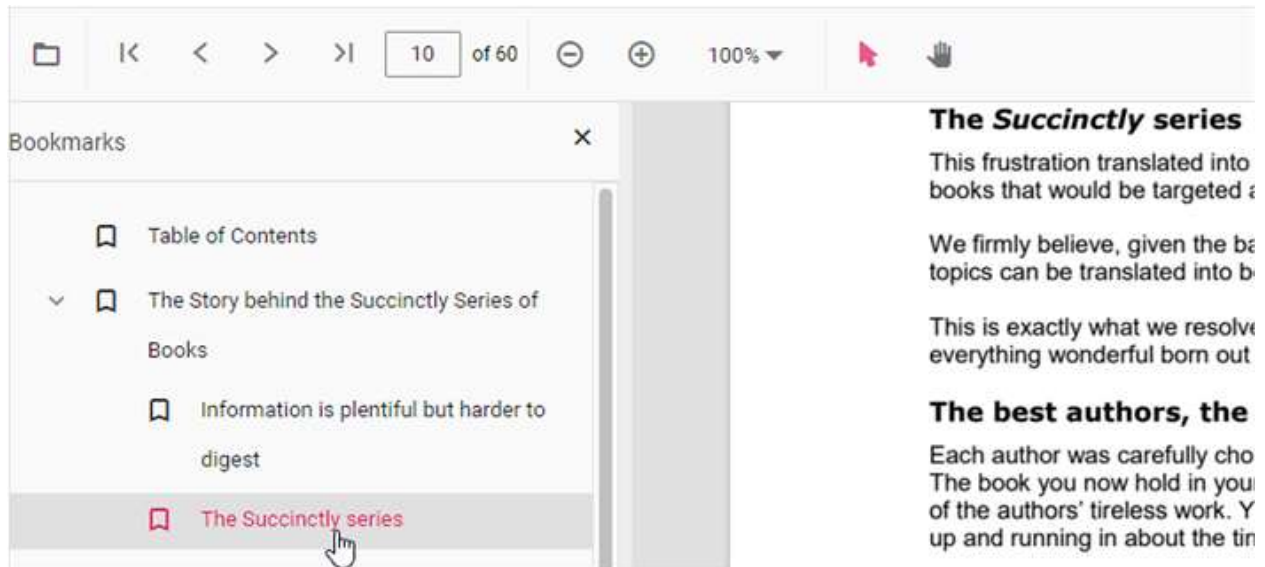
The Bookmarks saved in PDF files are loaded and made ready for easy navigation. You can enable/disable bookmark navigation by using the following code snippet.,

STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
enableBookmark: true,
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
enableBookmark: true,
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

To perform bookmark navigation, you can use the **goToBookmark** method. It's important to note that the **goToBookmark** method will throw an error if the specified bookmark does not exist in the PDF document.

Here is an example of how to use the **goToBookmark** method:

,

```
<button id="gotobookmark">Specfic Page</button>
```

,

```
`javascript
```

```
document.getElementById('gotobookmark').addEventListener('click', () => {
viewer.bookmark.goToBookmark(x, y);
});
```

,

x - Specifies the pageIndex for Navigate.

y - Specifies the Y coordinates value of the Page.

Also, you can use the **getBookmarks** method to retrieve a list of all the bookmarks in a PDF document. This method returns a List of Bookmark objects, which contain information about each bookmark.

Here is an example of how to use the **getBookmarks** method:

,

```
<button id="getBookmarks">retrieve bookmark</button>
```

,

```
`javascript
```

```
document.getElementById('getBookmarks').addEventListener('click', () => {
var getBookmarks = viewer.bookmark.getBookmarks();
```

```
console.log(getBookmarks)
```

```
});
```

```
,
```

Thumbnail navigation

Thumbnails is the miniature representation of actual pages in PDF files. This feature displays thumbnails of the pages and allows navigation.

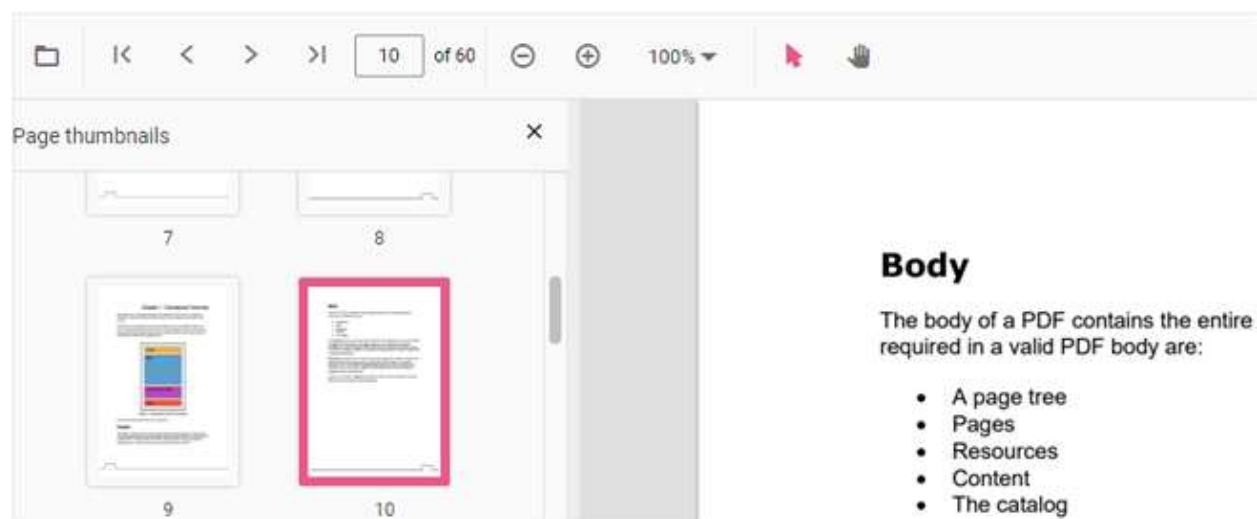
You can enable/disable thumbnail navigation by using the following code snippet.,

STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableThumbnail: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableThumbnail: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```



Hyperlink navigation

Hyperlink navigation features enables navigation to the URLs (website links) in a PDF file.



Table of content navigation

Table of contents navigation allows users to navigate to different parts of a PDF file that are listed in the table of contents section.

You can enable/disable link navigation by using the following code snippet.,

STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableHyperlink: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableHyperlink: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

You can change the open state of the hyperlink in the PDF Viewer by using the following code snippet,

STANDALONE

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableHyperlink: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
  hyperlinkOpenState: 'NewTab',
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');

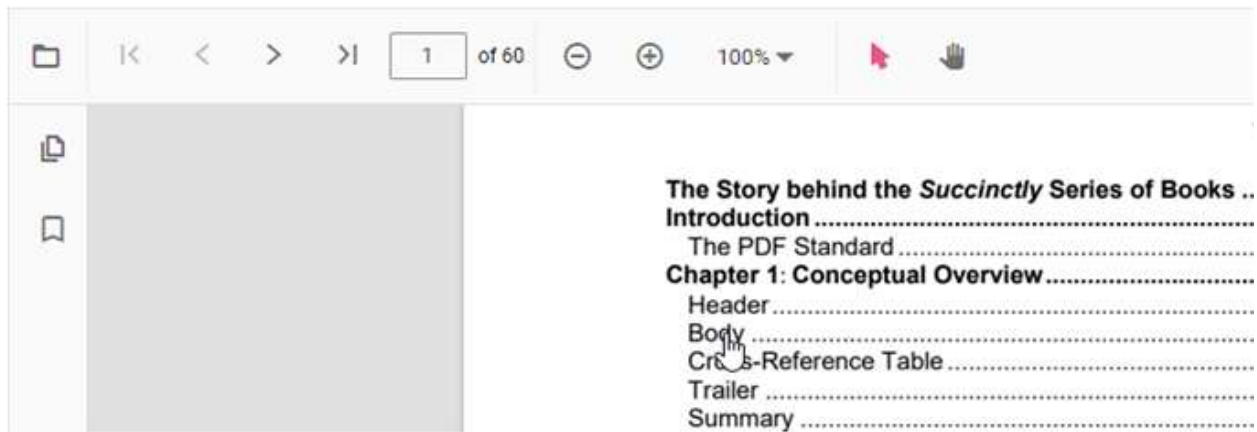
```

SERVER-BACKED

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableHyperlink: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
  hyperlinkOpenState: 'NewTab',
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');

```



See also

- [Toolbar items](#)
- [Feature Modules](#)

Magnification in EJ2 JavaScript Pdfviewer control

The magnification tools of the PDF Viewer contains ZoomIn, ZoomOut, Zoom, FitPage, and FitWidth tools in the default toolbar. The PDF Viewer also has an option to show or hide the magnification tools in the default toolbar.

The following code snippet describes how to enable the magnification in PDF Viewer.

```
`html
{% raw %}
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 fabric theme -->
<link href="{{:CDN_LINK}}ej2-pdfviewer/styles/fabric.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 PDF Viewer's global script -->
<script src="{{:CDN_LINK}}dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!--element which is going to render-->
<div id='container'>
<div id='PdfViewer' style="height:500px;width:100%;">
</div>
</div>
</body>
</html>
{% endraw %}
`
```

STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableMagnification: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

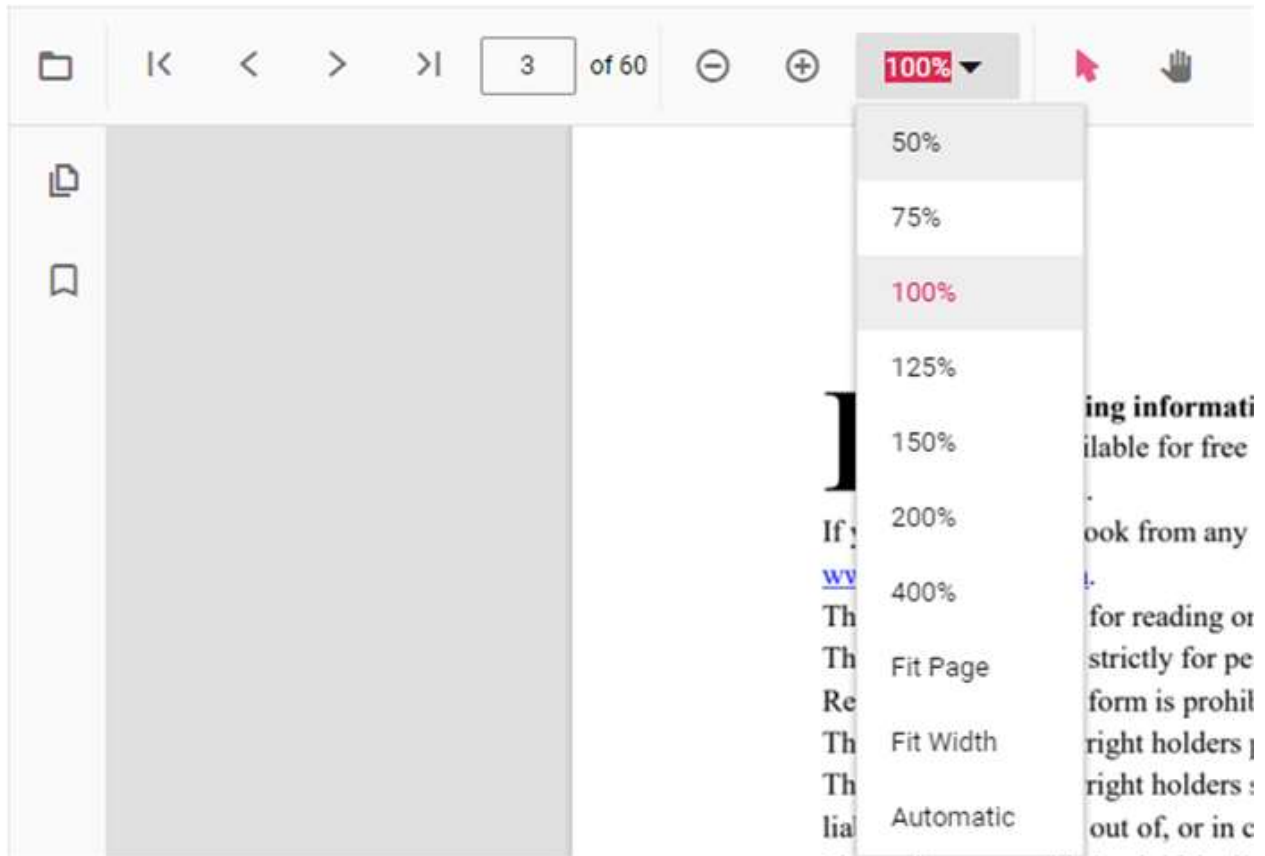
SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableMagnification: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
```

```
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

The following magnification options are available in the default toolbar of PDF Viewer,

- [ZoomIn](#):- Zoom in from the current zoom value of PDF pages.
- [ZoomOut](#):- Zoom out from the current zoom value of PDF pages.
- [Zoom](#):- Zoom to specific zoom value of PDF pages.
- [FitPage](#):- Fits the page width with in the available view port size.
- [FitWidth](#):- Fits the view port width based on the page content size.



PDF Viewer can support the zoom value ranges from 50 to 400.

See also

- [Toolbar items](#)
- [Feature Modules](#)

Accessibility in Syncfusion PDF Viewer components

The PDF Viewer component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the PDF Viewer component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

[WAI-ARIA](#) (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components. The following ARIA attributes are used in the PDF Viewer component:

Attributes	Purpose
---	---
<code>aria-disabled</code>	Indicates whether the PDF Viewer component is in a disabled state or not.
<code>aria-expanded</code>	Indicates whether the suggestion list has expanded or not.
<code>aria-readonly</code>	Indicates the readonly state of the PDF Viewer element.
<code>aria-haspopup</code>	Indicates whether the PDF Viewer input element has a suggestion list or not.
<code>aria-label</code>	Indicates the breadcrumb item text.
<code>aria-labelledby</code>	Provides a label for the PDF Viewer. Typically, the "aria-labelledby" attribute will contain the id of the element used as the PDF Viewer's title.
<code>aria-describedby</code>	This attribute points to the PDF Viewer element describing the one it's set on.
<code>aria-orientation</code>	Indicates whether the PDF Viewer element is oriented horizontally or vertically.
<code>aria-valuetext</code>	Returns the current text of the PDF Viewer.
<code>aria-valuemax</code>	Indicates the Maximum value of the PDF Viewer.
<code>aria-valuemin</code>	Indicates the Minimum value of the PDF Viewer.
<code>aria-valuenow</code>	Indicates the current value of the PDF Viewer.
<code>aria-controls</code>	Attribute is set to the button and it points to the corresponding content.

Keyboard interaction

The PDF Viewer component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Message component.

Press (Windows)	Press (Macintosh)	To do this
---	---	---
Shortcuts for page navigation		
Home	Function + Left arrow	Navigate to the first page
End	Function + Right arrow	Navigate to the last page
Up Arrow	Up Arrow	Navigate to the previous page
Down Arrow	Down Arrow	Navigate to the next page
Shortcuts for Zooming		
CONTROL + =	COMMAND + =	Perform zoom-in operation

| CONTROL + - | COMMAND + - | Perform zoom-out operation |

| CONTROL + 0 | COMMAND + 0 | Retain the zoom level to 1 |

|| **Shortcut for Text Search** |

| CONTROL + F | COMMAND + F | Open the search toolbar |

|| **Shortcut for Text Selection** |

| CONTROL + C | COMMAND + C | Copy the selected text or annotation or form field |

| CONTROL + X | COMMAND + X | Cut the selected text or annotation of the form field |

| CONTROL + Y | COMMAND + Y | Paste the selected text or annotation or form field |

|| **Shortcuts for the general operation** |

| CONTROL + Z | COMMAND + Z | Undo the action |

| CONTROL + Y | COMMAND + Y | Redo the action |

| CONTROL + P | COMMAND + P | Print the document |

| Delete | Delete | Delete the annotations and form fields |

STANDALONE

```
<html>
<head>
<!--Refer scripts and styles from CDN-->
<script
src="https://cdn.syncfusion.com/ej2/20.2.48/dist/ej2.min.js"
type="text/javascript">
</script>
<link
href="https://cdn.syncfusion.com/ej2/20.2.48/material.css"
rel="stylesheet"/>
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet"/>
<style>
body {
touch-action: none;
}
</style>
</head>
<body>
<div class="control-section">
<div class="content-wrapper">
<!--Add the PDF Viewer-->
<div id="pdfViewer" style="height: 640px; width: 100%"></div>
</div>
</div>
<script>
var viewer = new ej.pdfviewer.PdfViewer({
//Sets the document path for initial loading
documentPath: 'https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf',
resourceUrl: "https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
});
```

```
//Inject the dependencies required to render the PDF Viewer
ej.pdfviewer.PdfViewer.Inject(
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.BookmarkView,
ej.pdfviewer.ThumbnailView,
ej.pdfviewer.TextSelection, ej.pdfviewer.TextSearch, ej.pdfviewer.Print,
ej.pdfviewer.Navigation,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.Annotation,
ej.pdfviewer.FormFields, ej.pdfviewer.FormDesigner
);
viewer.appendTo('#pdfViewer');
</script>
</body>
</html>
```

SERVER-BACKED

```
<html>
<head>
<!--Refer scripts and styles from CDN-->
<script
src="https://cdn.syncfusion.com/ej2/20.2.48/dist/ej2.min.js"
type="text/javascript">
</script>
<link
href="https://cdn.syncfusion.com/ej2/20.2.48/material.css"
rel="stylesheet"/>
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet"/>
<style>
body {
touch-action: none;
}
</style>
</head>
<body>
<div class="control-section">
<div class="content-wrapper">
<!--Add the PDF Viewer-->
<div id="pdfViewer" style="height: 640px; width: 100%"></div>
</div>
</div>
<script>
var viewer = new ej.pdfviewer.PdfViewer({
//Sets the document path for initial loading
documentPath: 'https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf',
serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer',
});
//Inject the dependencies required to render the PDF Viewer
ej.pdfviewer.PdfViewer.Inject(
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.BookmarkView,
ej.pdfviewer.ThumbnailView,
ej.pdfviewer.TextSelection, ej.pdfviewer.TextSearch, ej.pdfviewer.Print,
ej.pdfviewer.Navigation,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.Annotation,
ej.pdfviewer.FormFields, ej.pdfviewer.FormDesigner
```

```
);  
viewer.appendTo('#pdfViewer');  
</script>  
</body>  
</html>
```

Ensuring accessibility

The PDF Viewer component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the PDF Viewer component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the PDF Viewer component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

Text search in EJ2 JavaScript Pdfviewer control

The Text Search option in PDF Viewer is used to find and highlight the text content from the document. You can enable/disable the text search using the following code snippet.

```
`html  
{% raw %}  
<!DOCTYPE html>  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<title>Essential JS 2</title>  
<!-- Essential JS 2 fabric theme -->  
<link href="{{CDN_LINK}}ej2-pdfviewer/styles/fabric.css" rel="stylesheet" type="text/css"/>  
<!-- Essential JS 2 PDF Viewer's global script -->  
<script src="{{CDN_LINK}}dist/ej2.min.js" type="text/javascript"></script>  
</head>  
<body>  
<!--element which is going to render-->  
<div id='container'>  
<div id='PdfViewer' style="height:500px;width:100%;">  
</div>  
</div>  
</body>  
</html>
```

```
{% endraw %}
```

```
,
```

STANDALONE

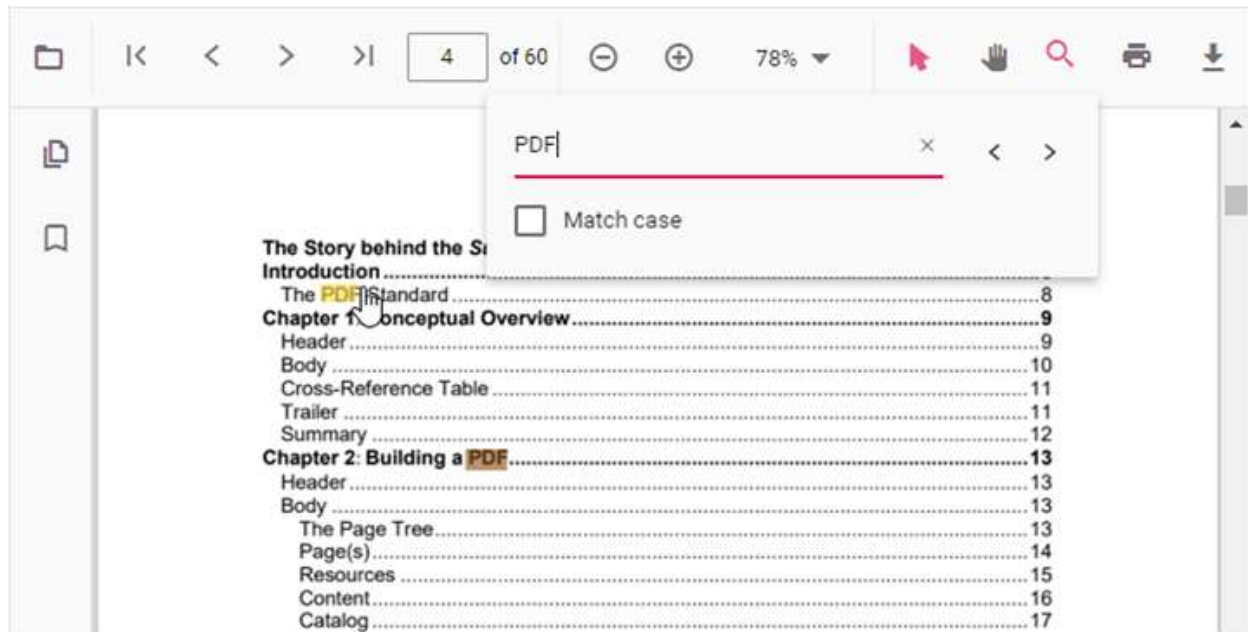
```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableTextSearch: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableTextSearch: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

The following text search methods are available in the PDF Viewer,

- [Search text](#):- Searches the target text in the PDF document and highlights the occurrences in the pages.
- [Search next](#):- Searches the next occurrence of the searched text from the current occurrence of the PdfViewer.
- [Search previous](#):- Searches the previous occurrence of the searched text from the current occurrence of the PdfViewer.
- [Cancel text search](#):- The text search can be cancelled and the highlighted occurrences from the PDF Viewer can be removed .



See also

- [Toolbar items](#)
- [Feature Modules](#)

Annotation

Interaction mode in EJ2 JavaScript Pdfviewer control

The PDF Viewer provides interaction mode for easy interaction with the loaded PDF document. Selection mode and panning mode are the two interactions modes.

Selection mode

In this mode, the text selection can be performed in the PDF document loaded in PDF Viewer. The panning and scrolling of the pages by touch cannot be performed in this mode. It allows users to select and copy text from the PDF files. This is helpful for copying and sharing text content. You can enable/disable the text selection using the following code snippet.

```
`html
{% raw %}
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 fabric theme -->
<link href="{{:CDN_LINK}}ej2-pdfviewer/styles/fabric.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 PDF Viewer's global script -->
<script src="{{:CDN_LINK}}dist/ej2.min.js" type="text/javascript"></script>
```

```
</head>
<body>
<!--element which is going to render-->
<div id='container'>
<div id='PdfViewer' style="height:500px;width:100%;">
</div>
</div>
</body>
</html>

{% enddraw %}
,
```

STANDALONE

```
import { PdfViewer, Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation,ThumbnailView,BookmarkView, TextSelection} from
'@syncfusion/ej2-pdfviewer';
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation,ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
let pdfviewer: PdfViewer = new PdfViewer({
enableTextSelection: true,
documentPath: 'https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf'
});
pdfviewer.appendTo('#PdfViewer');
```

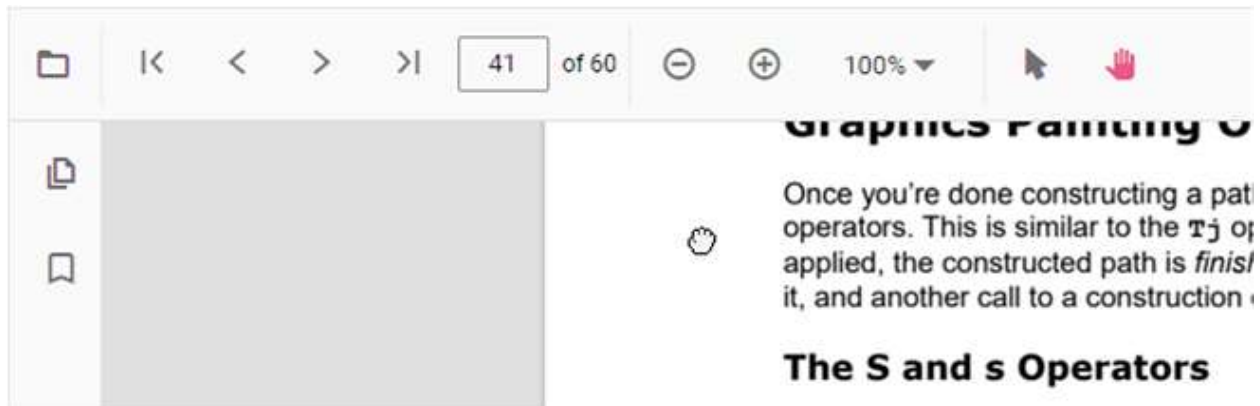
SERVER-BACKED

```
import { PdfViewer, Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation,ThumbnailView,BookmarkView, TextSelection} from
'@syncfusion/ej2-pdfviewer';
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation,ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
let pdfviewer: PdfViewer = new PdfViewer({
enableTextSelection: true,
documentPath: 'https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf',
serviceUrl : 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
pdfviewer.appendTo('#PdfViewer');
```



Panning Mode

In this mode, the panning and scrolling of the pages by touch can be performed in the PDF document loaded in the PDF Viewer, but the text selection cannot be performed.



You can switch the interaction mode of PDF Viewer by using the following code snippet.,

STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  interactionMode: 'Pan'
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  interactionMode: 'Pan'
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

See also

- [Toolbar items](#)
- [Feature Modules](#)

Form Designer

Create programmatically in EJ2 JavaScript Pdfviewer control

The PDF Viewer control provides the option to add, edit and delete the Form Fields. The Form Fields type supported by the PDF Viewer Control are:

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

Add a form field to PDF document programmatically

Using addFormField method, the form fields can be added to the PDF document programmatically. We need to pass two parameters in this method. They are Form Field Type and Properties of Form Field Type. To add form field programmatically, Use the following code.

INDEX.JS

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  documentPath: 'https://cdn.syncfusion.com/content/pdf/form-
designer.pdf',
  resourceUrl: 'https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-
lib'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.Annotation, ej.pdfviewer.FormDesigner,
ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
pdfviewer.documentLoad = function (args) {
  pdfviewer.formDesignerModule.addFormField("Textbox", { name: "First Name",
  bounds: { X: 146, Y: 229, Width: 150, Height: 24 } });
  pdfviewer.formDesignerModule.addFormField("Textbox", { name: "Middle Name",
  bounds: { X: 338, Y: 229, Width: 150, Height: 24 } });
  pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'Last
  Name',bounds: { X: 530, Y: 229, Width: 150, Height: 24 },});
  pdfviewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 148,
  Y: 289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
  pdfviewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 292,
  Y: 289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
```



```

pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOB
Month',bounds: { X: 146, Y: 320, Width: 35, Height: 24 },});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOB
Date',bounds: { X: 193, Y: 320, Width: 35, Height: 24 },});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOB
Year',bounds: { X: 242, Y: 320, Width: 35, Height: 24 },});
pdfviewer.formDesignerModule.addFormField('InitialField', {name:
'Agree',bounds: { X: 148, Y: 408, Width: 200, Height: 43 },});
pdfviewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
pdfviewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
pdfviewer.formDesignerModule.addFormField('CheckBox', {name: 'Email',bounds:
{ X: 242, Y: 664, Width: 20, Height: 20 },isChecked: false,});
pdfviewer.formDesignerModule.addFormField('CheckBox', {name: 'Appointment
Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
pdfviewer.formDesignerModule.addFormField('CheckBox', {name: 'Request for
Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20 },isChecked:
false,});
pdfviewer.formDesignerModule.addFormField('CheckBox', {name: 'Information
Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'My
Email',bounds: { X: 146, Y: 850, Width: 200, Height: 24 },});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'My
Phone',bounds: { X: 482, Y: 850, Width: 200, Height: 24 },});
pdfviewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOS
Month',bounds: { X: 386, Y: 923, Width: 35, Height: 24 },});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOS
Date',bounds: { X: 434, Y: 923, Width: 35, Height: 24 },});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOS
Year',bounds: { X: 482, Y: 923, Width: 35, Height: 24 },});
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 PDF Viewer</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript PDF Viewer Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pdfviewer/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

<!-- Essential JS 2 PDF Viewer's script -->
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="PdfViewer" style="height:500px;width:100%;"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To set up the **server-backed PDF Viewer**,

Add the below `serviceUrl` in the `index.html` file

`serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'`

Edit/Update form field programmatically

Using `updateFormField` method, Form Field can be updated programmatically. We should get the Form Field object/Id from `FormFieldCollections` property that you would like to edit and pass it as a parameter to `updateFormField` method. The second parameter should be the properties that you would like to update for Form Field programmatically. We have updated the value and background Color properties of Textbox Form Field.

INDEX.JS

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
    documentPath: 'https://cdn.syncfusion.com/content/pdf/form-designer.pdf',
    resourceUrl: 'https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-
lib'
});

```

```

ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
pdfviewer.documentLoad = function (args) {
    pdfviewer.formDesignerModule.addFormField("Textbox", { name: "First
Name", bounds: { X: 146, Y: 229, Width: 150, Height: 24 } });
    pdfviewer.formDesignerModule.addFormField("Textbox", { name: "Middle
Name", bounds: { X: 338, Y: 229, Width: 150, Height: 24 } });
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'Last
Name',bounds: { X: 530, Y: 229, Width: 150, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('RadioButton', {bounds: { X:
148, Y: 289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
    pdfviewer.formDesignerModule.addFormField('RadioButton', {bounds: { X:
292, Y: 289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOB
Month',bounds: { X: 146, Y: 320, Width: 35, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOB
Date',bounds: { X: 193, Y: 320, Width: 35, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOB
Year',bounds: { X: 242, Y: 320, Width: 35, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('InitialField', {name:
'Agree',bounds: { X: 148, Y: 408, Width: 200, Height: 43 },});
    pdfviewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
    pdfviewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
    pdfviewer.formDesignerModule.addFormField('CheckBox', {name:
'Email',bounds: { X: 242, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
    pdfviewer.formDesignerModule.addFormField('CheckBox', {name:
'Appointment Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20
},isChecked: false,});
    pdfviewer.formDesignerModule.addFormField('CheckBox', {name: 'Request
for Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20
},isChecked: false,});
    pdfviewer.formDesignerModule.addFormField('CheckBox', {name:
'Information Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20
},isChecked: false,});
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'My
Email',bounds: { X: 146, Y: 850, Width: 200, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'My
Phone',bounds: { X: 482, Y: 850, Width: 200, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOS
Month',bounds: { X: 386, Y: 923, Width: 35, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOS
Date',bounds: { X: 434, Y: 923, Width: 35, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOS
Year',bounds: { X: 482, Y: 923, Width: 35, Height: 24 },});

pdfviewer.formDesignerModule.updateFormField(pdfviewer.formFieldCollections[
0], { backgroundColor: 'red' } );

```

```
}

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 PDF Viewer</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript PDF Viewer Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pdfviewer/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <!-- Essential JS 2 PDF Viewer's script -->
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="PdfViewer" style="height:500px;width:100%;"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: To set up the **server-backed PDF Viewer**,

Add the below `serviceUrl` in the `index.html` file

`serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'`

Delete form field programmatically

Using `deleteFormField` method, the form field can be deleted programmatically. We should retrieve the Form Field object/Id from `FormFieldCollections` property that you would like to delete and pass it as a parameter to `deleteFormField` method. To delete a Form Field programmatically, use the following code.

INDEX.JS

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
    documentPath: 'https://cdn.syncfusion.com/content/pdf/form-designer.pdf',
    resourceUrl: 'https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-
lib'
});
ej.pdfviewer.PdfViewer.Inject( ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.Annotation, ej.pdfviewer.FormDesigner,
ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
pdfviewer.documentLoad = function (args) {
    pdfviewer.formDesignerModule.addFormField("Textbox", { name: "First
Name", bounds: { X: 146, Y: 229, Width: 150, Height: 24 } });
    pdfviewer.formDesignerModule.addFormField("Textbox", { name: "Middle
Name", bounds: { X: 338, Y: 229, Width: 150, Height: 24 } });
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'Last
Name',bounds: { X: 530, Y: 229, Width: 150, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('RadioButton', {bounds: { X:
148, Y: 289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
    pdfviewer.formDesignerModule.addFormField('RadioButton', {bounds: { X:
292, Y: 289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOB
Month',bounds: { X: 146, Y: 320, Width: 35, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOB
Date',bounds: { X: 193, Y: 320, Width: 35, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOB
Year',bounds: { X: 242, Y: 320, Width: 35, Height: 24 },});
    pdfviewer.formDesignerModule.addFormField('InitialField', {name:
'Agree',bounds: { X: 148, Y: 408, Width: 200, Height: 43 },});
    pdfviewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
    pdfviewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
    pdfviewer.formDesignerModule.addFormField('CheckBox', {name:
'Email',bounds: { X: 242, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
    pdfviewer.formDesignerModule.addFormField('CheckBox', {name:
'Appointment Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20
},isChecked: false,});
    pdfviewer.formDesignerModule.addFormField('CheckBox', {name: 'Request
for Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20
},isChecked: false,});
```

```

pdfviewer.formDesignerModule.addFormField('CheckBox', {name:
'Information Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20
},isChecked: false,});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'My
Email',bounds: { X: 146, Y: 850, Width: 200, Height: 24 },});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'My
Phone',bounds: { X: 482, Y: 850, Width: 200, Height: 24 },});
pdfviewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOS
Month',bounds: { X: 386, Y: 923, Width: 35, Height: 24 },});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOS
Date',bounds: { X: 434, Y: 923, Width: 35, Height: 24 },});
pdfviewer.formDesignerModule.addFormField('Textbox', {name: 'DOS
Year',bounds: { X: 482, Y: 923, Width: 35, Height: 24 },});

pdfviewer.formDesignerModule.deleteFormField(pdfviewer.formFieldCollections[
0]);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 PDF Viewer</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript PDF Viewer Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pdfviewer/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">

  <!-- Essential JS 2 PDF Viewer's script -->
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="PdfViewer" style="height:500px;width:100%;"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

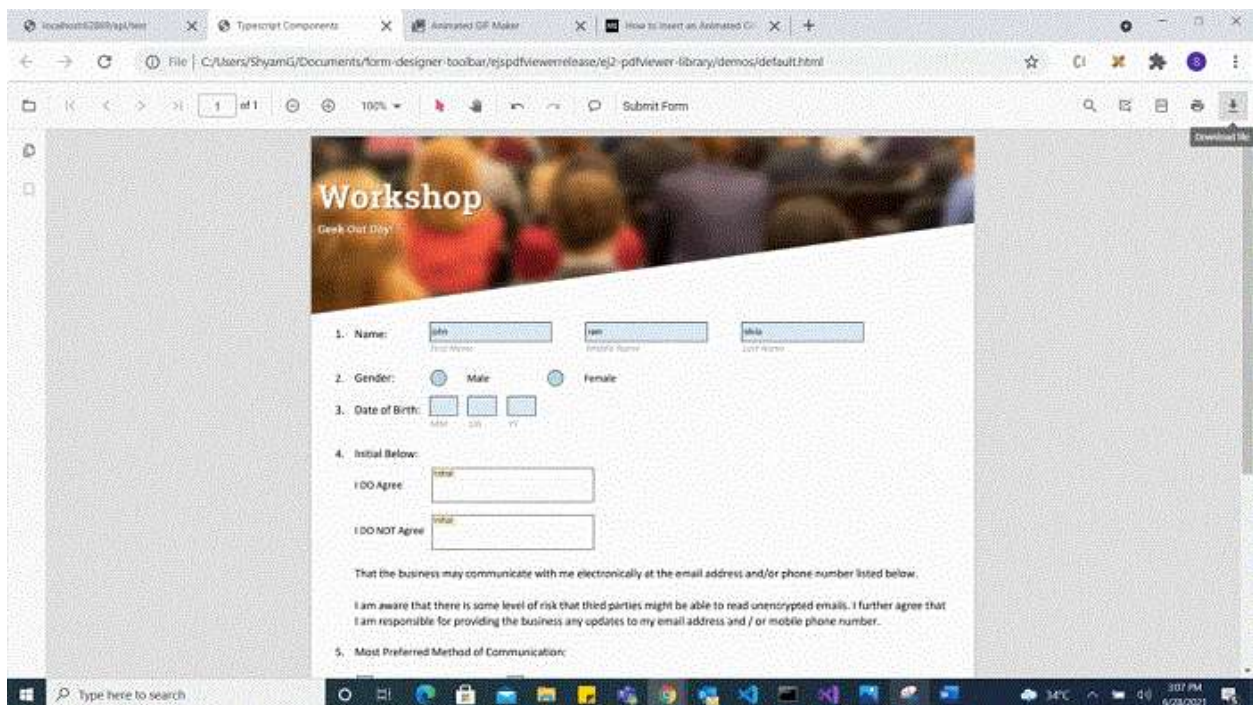
Note: To set up the **server-backed PDF Viewer**,

Add the below `serviceUrl` in the `index.html` file

`serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'`

Saving the form fields

When the download icon is selected on the toolbar, the Form Fields will be saved in the PDF document and this action will not affect the original document. Refer the below GIF for further reference.



You can invoke download action using following code snippet.

```
<button id="download">Download</button>
```

STANDALONE

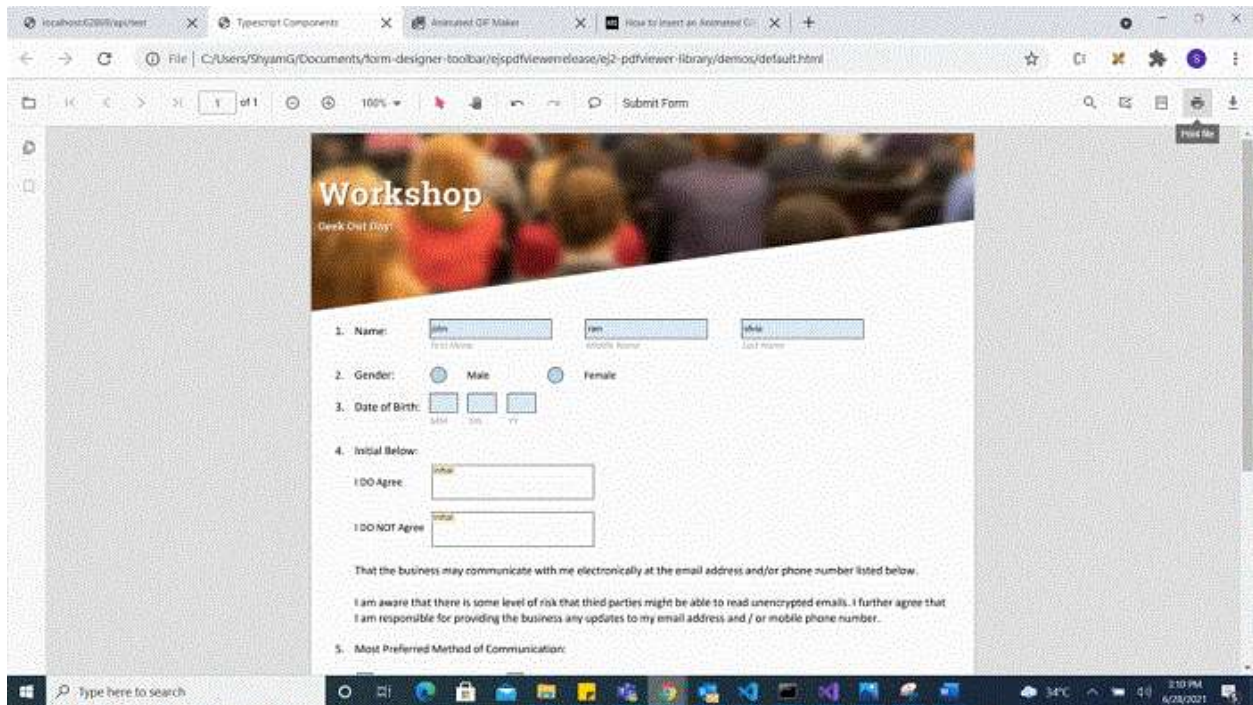
```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableDownload: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
document.getElementById('download').addEventListener('click', function () {
  pdfviewer.download()
});
```

SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enableDownload: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
document.getElementById('download').addEventListener('click', function () {
  pdfviewer.download()
});
```

Printing the form fields

When the print icon is selected on the toolbar, the PDF document will be printed along with the Form Fields added to the pages and this action will not affect the original document. Refer the below GIF for further reference.



You can invoke print action using the following code snippet.,

,

```
<button id="print">Print</button>
```

,

STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enablePrint: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
document.getElementById('print').addEventListener('click', function () {
  pdfviewer.print.print();
});
```

SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  enablePrint: true,
  documentPath: "https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
```

```
pdfviewer.appendTo('#PdfViewer');
document.getElementById('print').addEventListener('click', function () {
pdfviewer.print.print();
});
```

setFormFieldMode programmatically

The **setFormFieldMode** method is a function in the Syncfusion PDF Viewer library that allows you to add a form field dynamically by passing the type of the form field. You can pass the form fields as a parameter like below.

```
<button id="addPasswordField">Add Password Field</button>
```

STANDALONE

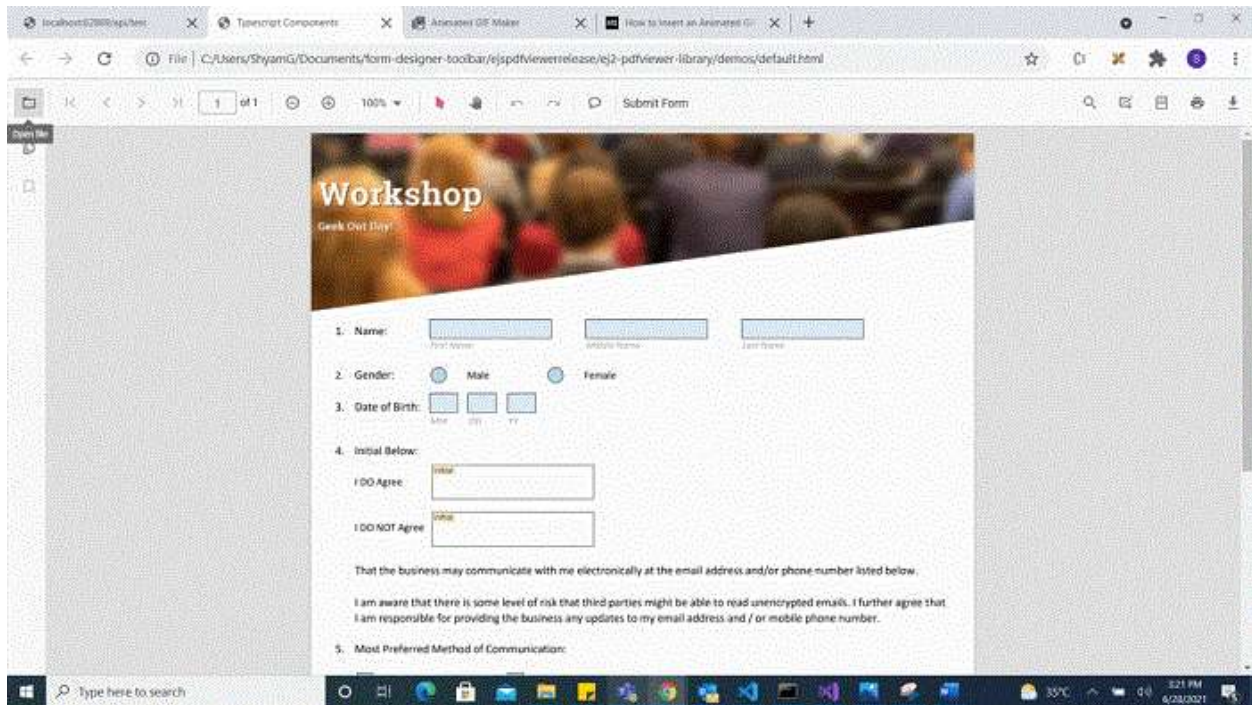
```
var pdfviewer = new ej.pdfviewer.PdfViewer({
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
document.getElementById('addPasswordField').addEventListener('click',
function () {
pdfviewer.formDesignerModule.setFormFieldMode("Password");
});
```

SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
document.getElementById('addPasswordField').addEventListener('click',
function () {
pdfviewer.formDesignerModule.setFormFieldMode("Password");
});
```

Open the existing PDF document

We can open the already saved PDF document contains Form Fields in it by clicking the open icon in the toolbar. Refer the below GIF for further reference.



Validate form fields

The form fields in the PDF Document will be validated when the `enableFormFieldsValidation` is set to `true` and hook the `validateFormFields`. The `validateFormFields` will be triggered when the PDF document is downloaded or printed with the non-filled form fields. The non-filled fields will be obtained in the `nonFillableFields` property of the event arguments of `validateFormFields`.

Add the following code snippet to validate the form fields,

STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  documentPath: "https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
viewer.enableFormFieldsValidation = true;
viewer.validateFormFields = function (args) {
  var nonfilledFormFields = args.nonFillableFields;
};
```

SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  documentPath: "https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
```

```
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
viewer.enableFormFieldsValidation = true;
viewer.validateFormFields = function (args) {
  var nonfilledFormFields = args.nonFillableFields;
};
```

Export and import form fields

The PDF Viewer control provides the support to export and import the form field data in the following formats using the `importFormFields`, `exportFormFields`, and `exportFormFieldsAsObject` methods.

- FDF
- XFDF
- JSON

Export and import as FDF

Using the `exportFormFields` method, the form field data can be exported in the specified data format. This method accepts two parameters:

- The first one must be the destination path for the exported data. If the path is not specified, it will ask for the location while exporting.
- The second parameter should be the format type of the form data.

The following code explains how to export the form field data as FDF.

```
`javascript
<button id="exportFdf">Export FDF</button>
<button id="importFdf">Import FDF</button>
<!--Add the PDF Viewer-->
<div id="pdfViewer" style="height: 640px; width: 100%"></div>
// Event triggers on Export FDF button click.
document.getElementById('exportFdf').addEventListener('click', ()=> {
// Data must be the desired path for the exported document.
viewer.exportFormFields('Data', FormFieldDataFormat.Fdf);
});
// Event triggers on Import FDF button click.
document.getElementById('importFdf').addEventListener('click', ()=> {
// The file for importing the form fields should be placed in the desired location, and the path should be
provided correctly.
viewer.importFormFields('File', FormFieldDataFormat.Fdf);
});
```

`

Export and import as XFDF

The following code explains how to export the form field data as XFDF.

```
`javascript
<button id="exportXfdf">Export XFDF</button>
<button id="importXfdf">Import XFDF</button>
<!--Add the PDF Viewer-->
<div id="pdfViewer" style="height: 640px; width: 100%"></div>
// Event triggers on Export XFDF button click.
document.getElementById('exportXfdf').addEventListener('click', ()=> {
// Data must be the desired path for the exported document.
viewer.exportFormFields('Data', FormFieldDataFormat.Xfdf);
});
// Event triggers on Import XFDF button click.
document.getElementById('importXfdf').addEventListener('click', ()=> {
// The file for importing the form fields should be placed in the desired location, and the path should be
provided correctly.
viewer.importFormFields('File', FormFieldDataFormat.Xfdf);
});
```

`

Export and import as JSON

The following code explains how to export the form field data as JSON.

```
`javascript
<button id="exportJson">Export JSON</button>
<button id="importJson">Import JSON</button>
<!--Add the PDF Viewer-->
<div id="pdfViewer" style="height: 640px; width: 100%"></div>
// Event triggers on Export JSON button click.
document.getElementById('exportJson').addEventListener('click', ()=> {
// Data must be the desired path for the exported document.
viewer.exportFormFields('Data', FormFieldDataFormat.Json);
});
// Event triggers on Import JSON button click.
document.getElementById('importJson').addEventListener('click', ()=> {
```

// The file for importing the form fields should be placed in the desired location, and the path should be provided correctly

```
viewer.importFormFields('File', FormFieldDataFormat.Json);
});
```

Export and import as Object

The PDF Viewer control supports exporting the form field data as an object, and the exported data will be imported into the current PDF document from the object.

The following code shows how to export the form field data as an object and import the form field data from that object into the current PDF document via a button click.

```
`javascript
<button id="exportDataAsObject">Export Object</button>
<button id="importData">Import Data</button>
<!--Add the PDF Viewer-->
<div id="pdfViewer" style="height: 640px; width: 100%"></div>
var exportedData;
// Event triggers on Export Object button click.
document.getElementById('exportDataAsObject').addEventListener('click', ()=> {
// Export the form field data to an FDF object.
viewer.exportFormFieldsAsObject(FormFieldDataFormat.Fdf).then(value => {
exportedData = value;
});
// // Export the form field data to an XFDF object.
// viewer.exportFormFieldsAsObject(FormFieldDataFormat.Xfdf).then(value =>{
//   exportedData = value;
// })
// // Export the form field data to an JSON object.
// viewer.exportFormFieldsAsObject(FormFieldDataFormat.Json).then(value =>{
//   exportedData = value;
// })
});
// Event triggers on Import Data button click.
document.getElementById('importData').addEventListener('click', ()=> {
// Import the form field data from the FDF object into the current PDF document.
```



```
viewer.importFormFields(exportedData, FormFieldDataFormat.Fdf);
//// Import the form field data from the XFDF object into the current PDF document.
//viewer.importFormFields (exportedData, FormFieldDataFormat.Xfdf);
//// Import the form field data from the FDF object into the current PDF document.
//viewer.importFormFields (exportedData, FormFieldDataFormat.Json);
});
`
```

Form field properties

Form field properties in Syncfusion PDF Viewer allow you to customize and interact with form fields embedded within PDF documents. This documentation provides an overview of the form field properties supported by the Syncfusion PDF Viewer and explains how to use them effectively.

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

Signature and initial fields settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the signature field properties on a button click.

```
`html
<button id="updateProperties">Update Properties</button>
`

`javascript
document.getElementById('updateProperties').addEventListener('click',function() {
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Initial',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Initial',
thickness: 4
});
});
`
```

```
});
});
,
```

The following code example explains how to update the properties of the signature field added to the document from the form designer toolbar.

STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the signature field settings
pdfviewer.signatureFieldSettings = {
  // Set the name of the form field element.
  name: 'Signature',
  // Specify whether the signature field is in read-only or read-write mode.
  isReadOnly: false,
  // Set the visibility of the form field.
  visibility: 'visible',
  // Specify whether the field is mandatory or not.
  isRequired: false,
  // Specify whether to print the signature field.
  isPrint: true,
  // Set the text to be displayed as a tooltip.
  tooltip: 'Signature',
  // Set the thickness of the Signature field. To hide the borders, set the
  value to 0 (zero).
  thickness: 4,
  // Specify the properties of the signature Dialog Settings in the signature
  field.
  signatureDialogSettings: {
    displayMode: ej.pdfviewer.DisplayMode.Draw | ej.pdfviewer.DisplayMode.Upload
    | ej.pdfviewer.DisplayMode.Text,
    hideSaveSignature: false,
  },
  // Specify the properties of the signature indicator in the signature field.
  signatureIndicatorSettings: {
    opacity: 1,
    backgroundColor: '#237ba2',
    height: 50,
    fontSize: 15,
    text: 'Signature Field',
    color: 'white'
  },
};
```

SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
```



```

documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the signature field settings
pdfviewer.signatureFieldSettings = {
// Set the name of the form field element.
name: 'Signature',
// Specify whether the signature field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the signature field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Signature',
// Set the thickness of the Signature field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specify the properties of the signature Dialog Settings in the signature
field.
signatureDialogSettings: {
displayMode: ej.pdfviewer.DisplayMode.Draw | ej.pdfviewer.DisplayMode.Upload
| ej.pdfviewer.DisplayMode.Text,
hideSaveSignature: false,
},
// Specify the properties of the signature indicator in the signature field.
signatureIndicatorSettings: {
opacity: 1,
backgroundColor: '#237ba2',
height: 50,
fontSize: 15,
text: 'Signature Field',
color: 'white'
},
};

```



The following code example explains how to update the properties of the initial field added to the document from the form designer toolbar.

STANDALONE

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",

```

```

});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the signature field settings
pdfviewer.initialFieldSettings = {
// Set the name of the form field element.
name: 'Initial',
// Specify whether the initial field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the initial field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Initial',
// Set the thickness of the initial field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specifies the properties of the initial indicator in the initial field.
initialIndicatorSettings: {
opacity: 1,
backgroundColor: '#237ba2',
height: 50,
fontSize: 15,
text: 'Initial Field',
color: 'white'
},
// Specify the properties of the initial Dialog Settings in the initial
field.
initialDialogSettings: {
displayMode: ej.pdfviewer.DisplayMode.Draw | ej.pdfviewer.DisplayMode.Upload
| ej.pdfviewer.DisplayMode.Text,
hideSaveSignature: false
}
};

```

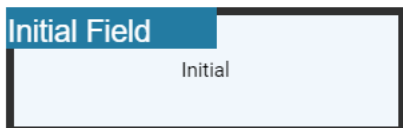
SERVER-BACKED

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the signature field settings
pdfviewer.initialFieldSettings = {
// Set the name of the form field element.
name: 'Initial',

```

```
// Specify whether the initial field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the initial field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Initial',
// Set the thickness of the initial field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specifies the properties of the initial indicator in the initial field.
initialIndicatorSettings: {
  opacity: 1,
  backgroundColor: '#237ba2',
  height: 50,
  fontSize: 15,
  text: 'Initial Field',
  color: 'white'
},
// Specify the properties of the initial Dialog Settings in the initial
field.
initialDialogSettings: {
  displayMode: ej.pdfviewer.DisplayMode.Draw | ej.pdfviewer.DisplayMode.Upload
| ej.pdfviewer.DisplayMode.Text,
  hideSaveSignature: false
}
};
```



Textbox field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the Textbox field properties on a button click.

`html

```
<button id="updateProperties">Update Properties</button>
```

,

`javascript

```
document.getElementById('updateProperties').addEventListener('click',function() {
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Textbox',
```

```

isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Textbox',
thickness: 4,
value: 'Textbox',
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: 'white',
alignment: 'Left',
maxLength: 0,
isMultiline: false,
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }
});
});
`

```

The following code example explains how to update the properties of the Textbox field added to the document from the form designer toolbar.

STANDALONE

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the Textbox field settings
pdfviewer.textFieldSettings = {
  // Set the name of the form field element.
  name: 'Textbox',
  // Specify whether the Textbox field is in read-only or read-write mode.
  isReadOnly: false,
  // Set the visibility of the form field.
  visibility: 'visible',
  // Specify whether the field is mandatory or not.
  isRequired: false,

```

```

// Specify whether to print the Textbox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Textbox',
// Set the thickness of the Textbox field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'Textbox',
// Set the font family of the textbox field.
fontFamily: 'Courier',
// Set the font size of the textbox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the textbox field.
color: 'black',
// Set the border color of the textbox field.
borderColor: 'black',
// Set the background color of the textbox field.
backgroundColor: 'White',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
// Allows multiline input in the text field. FALSE, by default.
isMultiline: false
};

```

SERVER-BACKED

```

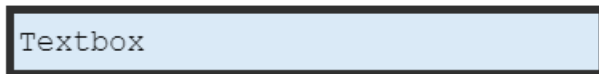
var pdfviewer = new ej.pdfviewer.PdfViewer({
  documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the Textbox field settings
pdfviewer.textFieldSettings = {
  // Set the name of the form field element.
  name: 'Textbox',
  // Specify whether the Textbox field is in read-only or read-write mode.
  isReadOnly: false,
  // Set the visibility of the form field.
  visibility: 'visible',
  // Specify whether the field is mandatory or not.
  isRequired: false,
  // Specify whether to print the Textbox field.
  isPrint: true,
  // Set the text to be displayed as a tooltip.
  tooltip: 'Textbox',
  // Set the thickness of the Textbox field. To hide the borders, set the
value to 0 (zero).

```

```

thickness: 4,
// Set the value of the form field element.
value: 'Textbox',
// Set the font family of the textbox field.
fontFamily: 'Courier',
// Set the font size of the textbox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the textbox field.
color: 'black',
// Set the border color of the textbox field.
borderColor: 'black',
// Set the background color of the textbox field.
backgroundColor: 'White',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
// Allows multiline input in the text field. FALSE, by default.
isMultiline: false
};

```



Password field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the Password field properties on a button click.

``html`

```
<button id="updateProperties">Update Properties</button>
```

```

``javascript`

```

document.getElementById('updateProperties').addEventListener('click',function() {
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Password',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Password',

```

```

thickness: 4,
value:'Password',
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: 'white',
alignment: 'Left',
maxLength: 0,
bounds: { X: 148, Y: 229, Width: 150, Height: 24 }
});
});
`

```

The following code example explains how to update the properties of the Password field added to the document from the form designer toolbar.

#### **STANDALONE**

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the Password field settings
pdfviewer.passwordFieldSettings = {
 // Set the name of the form field element.
 name: 'Password',
 // Specify whether the Password field is in read-only or read-write mode.
 isReadOnly: false,
 // Set the visibility of the form field.
 visibility: 'visible',
 // Specify whether the field is mandatory or not.
 isRequired: false,
 // Specify whether to print the Password field.
 isPrint: true,
 // Set the text to be displayed as a tooltip.
 tooltip: 'Password',
 // Set the thickness of the Password field. To hide the borders, set the
 value to 0 (zero).
 thickness: 4,
 // Set the value of the form field element.
 value: 'Password',
 // Set the font family of the Password field.

```

```

fontFamily: 'Courier',
// Set the font size of the Password field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the Password field.
color: 'black',
// Set the border color of the Password field.
borderColor: 'black',
// Set the background color of the Password field.
backgroundColor: 'white',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
};

```

### **SERVER-BACKED**

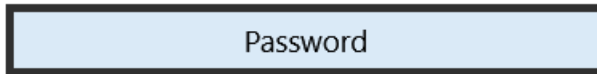
```

var pdfviewer = new ej.pdfviewer.PdfViewer({
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the Password field settings
pdfviewer.passwordFieldSettings = {
// Set the name of the form field element.
name: 'Password',
// Specify whether the Password field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the Password field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Password',
// Set the thickness of the Password field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'Password',
// Set the font family of the Password field.
fontFamily: 'Courier',
// Set the font size of the Password field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the Password field.
color: 'black',
// Set the border color of the Password field.
};

```



```
borderColor: 'black',
// Set the background color of the Password field.
backgroundColor: 'white',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
};
```



#### CheckBox field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the CheckBox field properties on a button click.

`html

```
<button id="updateProperties">Update Properties</button>
```

,

`javascript

```
document.getElementById('updateProperties').addEventListener('click',function() {
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'CheckBox',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'CheckBox',
thickness: 4,
isChecked: true,
backgroundColor: 'white',
borderColor: 'black',
value: 'CheckBox',
});
});
,
```

The following code example explains how to update the properties of the CheckBox field added to the document from the form designer toolbar.

### **STANDALONE**

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the CheckBox field settings
pdfviewer.checkBoxFieldSettings = {
 // Set the name of the form field element.
 name: 'CheckBox',
 // Specify whether the CheckBox field is in read-only or read-write mode.
 isReadOnly: false,
 // Set the visibility of the form field.
 visibility: 'visible',
 // Specify whether the field is mandatory or not.
 isRequired: false,
 // Specify whether to print the CheckBox field.
 isPrint: true,
 // Set the text to be displayed as a tooltip.
 tooltip: 'CheckBox',
 // Set the thickness of the CheckBox field. To hide the borders, set the
 // value to 0 (zero).
 thickness: 4,
 // Specifies whether the check box is in checked state or not.
 isChecked: true,
 // Set the background color of the check box in hexadecimal string format.
 backgroundColor: 'white',
 // Set the border color of the check box field.
 borderColor: 'black',
 // Set the value of
};
```

### **SERVER-BACKED**

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
 serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the CheckBox field settings
pdfviewer.checkBoxFieldSettings = {
 // Set the name of the form field element.
 name: 'CheckBox',
 // Specify whether the CheckBox field is in read-only or read-write mode.
 isReadOnly: false,
```

```
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the CheckBox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'CheckBox',
// Set the thickness of the CheckBox field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specifies whether the check box is in checked state or not.
isChecked: true,
// Set the background color of the check box in hexadecimal string format.
backgroundColor: 'white',
// Set the border color of the check box field.
borderColor: 'black',
// Set the value of
};
```



#### RadioButton field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the RadioButton field properties on a button click.

`html

```
<button id="updateProperties">Update Properties</button>
```

,

`javascript

```
document.getElementById('updateProperties').addEventListener('click',function() {
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'RadioButton',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'RadioButton',
```

```

thickness: 4,
isSelected: true,
backgroundColor: 'white',
borderColor: 'black',
value: 'RadioButton'
});
});
`

```

The following code example explains how to update the properties of the RadioButton field added to the document from the form designer toolbar.

### **STANDALONE**

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the RadioButton field settings
pdfviewer.radioButtonFieldSettings = {
 // Set the name of the form field element.
 name: 'RadioButton',
 // Specify whether the RadioButton field is in read-only or read-write mode.
 isReadOnly: false,
 // Set the visibility of the form field.
 visibility: 'visible',
 // Specify whether the field is mandatory or not.
 isRequired: false,
 // Specify whether to print the RadioButton field.
 isPrint: true,
 // Set the text to be displayed as a tooltip.
 tooltip: 'RadioButton',
 // Set the thickness of the RadioButton field. To hide the borders, set the
 // value to 0 (zero).
 thickness: 4,
 // Specifies whether the radio button is in checked state or not.
 isSelected: true,
 // Set the background color of the radio button in hexadecimal string
 // format.
 backgroundColor: 'white',
 // Set the border color of the radio button field.
 borderColor: 'black',
 // Set the value of the form field element.
 value: 'RadioButton'
};

```

### **SERVER-BACKED**

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
 serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
// Properties to customize the RadioButton field settings
pdfviewer.radioButtonFieldSettings = {
 // Set the name of the form field element.
 name: 'RadioButton',
 // Specify whether the RadioButton field is in read-only or read-write mode.
 isReadOnly: false,
 // Set the visibility of the form field.
 visibility: 'visible',
 // Specify whether the field is mandatory or not.
 isRequired: false,
 // Specify whether to print the RadioButton field.
 isPrint: true,
 // Set the text to be displayed as a tooltip.
 tooltip: 'RadioButton',
 // Set the thickness of the RadioButton field. To hide the borders, set the
 // value to 0 (zero).
 thickness: 4,
 // Specifies whether the radio button is in checked state or not.
 isSelected: true,
 // Set the background color of the radio button in hexadecimal string
 // format.
 backgroundColor: 'white',
 // Set the border color of the radio button field.
 borderColor: 'black',
 // Set the value of the form field element.
 value: 'RadioButton'
};

```



#### ListBox field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the ListBox field properties on a button click.

`html

```
<button id="updateProperties">Update Properties</button>
```

`

`javascript

```
document.getElementById('updateProperties').addEventListener('click',function() {
```

```

var formField = viewer.retrieveFormFields();

var customOptions = [{itemName:'item1',itemValue:'item1'}, {itemName:'item2',itemValue:'item2'},
{itemName:'item3',itemValue:'item3'}]

viewer.formDesignerModule.updateFormField(formField[0], {
name: 'ListBox',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'ListBox',
thickness: 4,
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: 'white',
alignment: 'Left',
options: customOptions,
});
});
`

```

The following code example explains how to update the properties of the Listbox field added to the document from the form designer toolbar.

#### **STANDALONE**

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
var customOptions = [{itemName:'item1',itemValue:'item1'},
{itemName:'item2',itemValue:'item2'}, {itemName:'item3',itemValue:'item3'}]
// Properties to customize the Listbox field settings
pdfviewer.listBoxFieldSettings = {
// Set the name of the form field element.
name: 'ListBox',
// Specify whether the Listbox field is in read-only or read-write mode.

```

```

isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the ListBox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'ListBox',
// Set the thickness of the ListBox field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'ListBox',
// Set the font family of the ListBox field.
fontFamily: 'Courier',
// Set the font size of the ListBox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the ListBox field.
color: 'black',
// Set the border color of the ListBox field.
borderColor: 'black',
// Set the background color of the ListBox field.
backgroundColor: 'White',
// Set the alignment of the text.
alignment: 'Left',
// Set the listbox items.
options: customOptions
};

```

### **SERVER-BACKED**

```

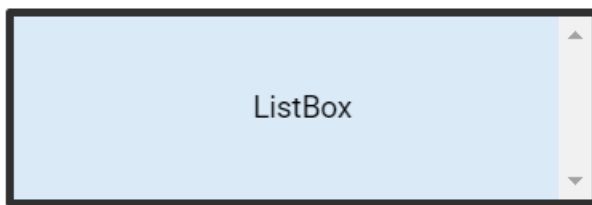
var pdfviewer = new ej.pdfviewer.PdfViewer({
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
var customOptions = [{itemName:'item1',itemValue:'item1'},
{itemName:'item2',itemValue:'item2'}, {itemName:'item3',itemValue:'item3'}]
// Properties to customize the Listbox field settings
pdfviewer.listBoxFieldSettings = {
// Set the name of the form field element.
name: 'ListBox',
// Specify whether the ListBox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the ListBox field.

```

```

isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'ListBox',
// Set the thickness of the ListBox field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'ListBox',
// Set the font family of the ListBox field.
fontFamily: 'Courier',
// Set the font size of the ListBox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the ListBox field.
color: 'black',
// Set the border color of the ListBox field.
borderColor: 'black',
// Set the background color of the ListBox field.
backgroundColor: 'White',
// Set the alignment of the text.
alignment: 'Left',
// Set the listbox items.
options: customOptions
};

```



#### DropDown field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the DropDown field properties on a button click.

``html`

```
<button id="updateProperties">Update Properties</button>
```

`,`

``javascript`

```

document.getElementById('updateProperties').addEventListener('click',function() {
var formField = viewer.retrieveFormFields();

var customOptions = [{itemName:'item1',itemValue:'item1'}, {itemName:'item2',itemValue:'item2'},
{itemName:'item3',itemValue:'item3'}]

viewer.formDesignerModule.updateFormField(formField[0], {

```



```

name: 'DropDown',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'DropDown',
thickness: 4,
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: 'white',
alignment: 'Left',
options: customOptions,
});
});
`

```

The following code example explains how to update the properties of the Dropdown field added to the document from the form designer toolbar.

### **STANDALONE**

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
var customOptions = [{itemName:'item1',itemValue:'item1'},
{itemName:'item2',itemValue:'item2'}, {itemName:'item3',itemValue:'item3'}]
// Properties to customize the DropDown field settings
pdfviewer.DropDownFieldSettings = {
// Set the name of the form field element.
name: 'DropDown',
// Specify whether the DropDown field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the DropDown field.

```

```

isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'DropDown',
// Set the thickness of the DropDown field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'DropDown',
// Set the font family of the DropDown field.
fontFamily: 'Courier',
// Set the font size of the DropDown field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the DropDown field.
color: 'black',
// Set the border color of the DropDown field.
borderColor: 'black',
// Set the background color of the DropDown field.
backgroundColor: 'White',
// Set the alignment of the text.
alignment: 'Left',
// Set the DropDown items.
options: customOptions
};

```

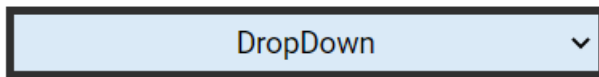
### **SERVER-BACKED**

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
var customOptions = [{itemName:'item1',itemValue:'item1'},
{itemName:'item2',itemValue:'item2'}, {itemName:'item3',itemValue:'item3'}]
// Properties to customize the DropDown field settings
pdfviewer.DropDownFieldSettings = {
// Set the name of the form field element.
name: 'DropDown',
// Specify whether the DropDown field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the DropDown field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'DropDown',
// Set the thickness of the DropDown field. To hide the borders, set the
value to 0 (zero).
thickness: 4,

```

```
// Set the value of the form field element.
value: 'DropDown',
// Set the font family of the DropDown field.
fontFamily: 'Courier',
// Set the font size of the DropDown field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the DropDown field.
color: 'black',
// Set the border color of the DropDown field.
borderColor: 'black',
// Set the background color of the DropDown field.
backgroundColor: 'White',
// Set the alignment of the text.
alignment: 'Left',
// Set the DropDown items.
options: customOptions
};
```



### Create with user interface interaction in EJ2 JavaScript Pdfviewer control

The PDF viewer control provides the option for interaction with Form Fields such as Drag and resize. you can draw a Form Field dynamically by clicking the Form Field icon on the toolbar and draw it in the PDF document. The Form Fields type supported by the PDF Viewer Control are:

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

### Enable or Disable form designer toolbar

We should inject FormDesigner module and set enableFormDesignerToolbar as true to enable the Form designer icon on the toolbar. By default, enableFormDesignerToolbar is set as true. Use the following code to inject FormDesigner module and to enable the enableFormDesignerToolbar property.

### STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
 documentPath: "https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
 enableFormDesignerToolbar: true
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
```

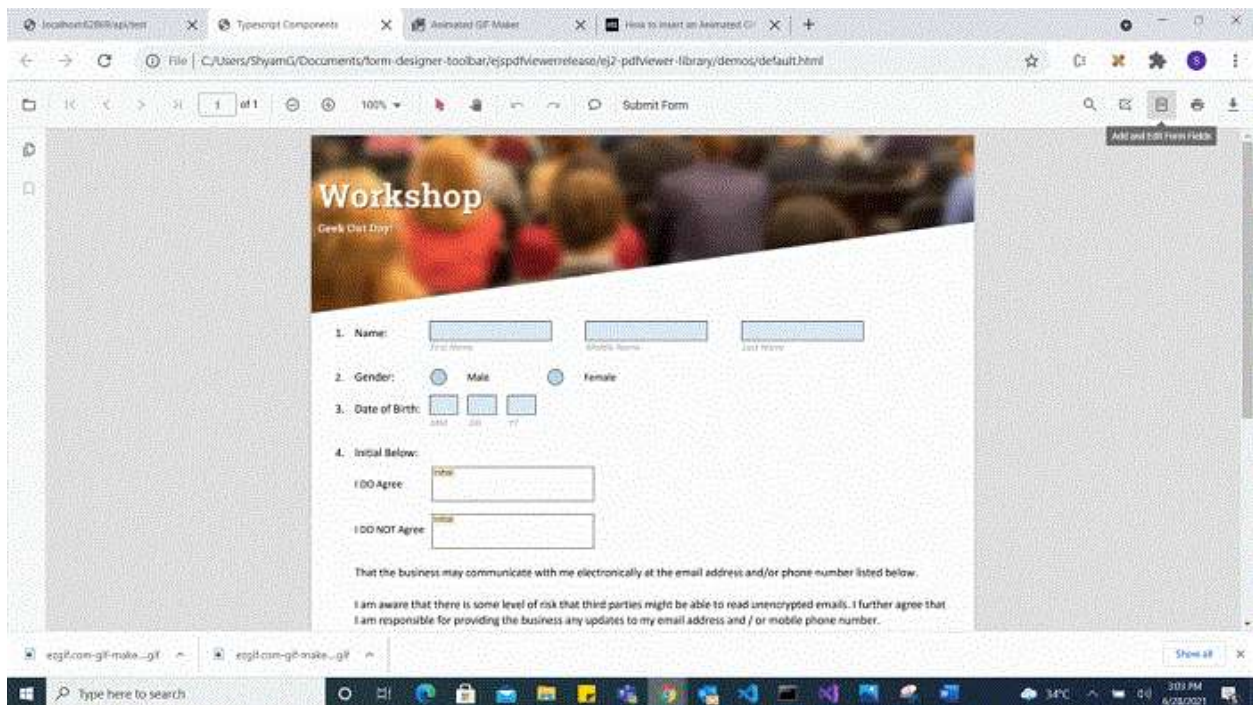
```
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
```

## SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
 documentPath: "https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
 serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer',
 enableFormDesignerToolbar: true
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
```

### Add the form field dynamically

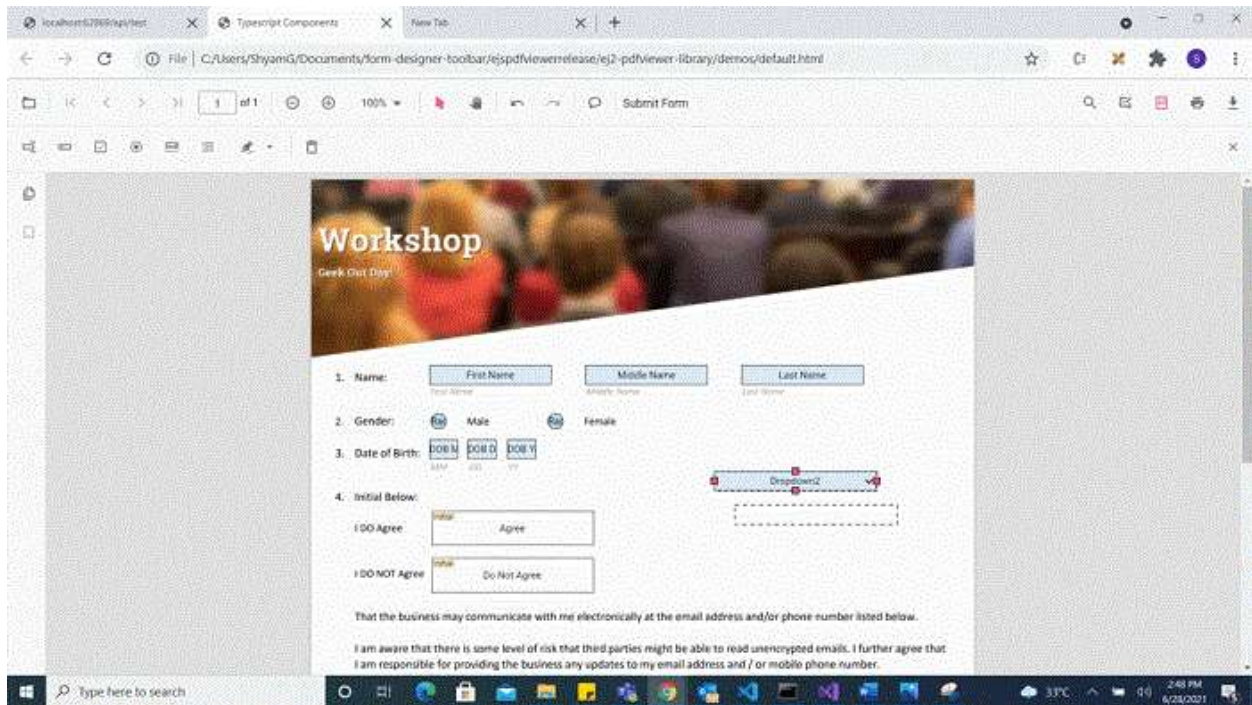
Click the Form Field icon on the toolbar and then click on to the PDF document to draw a Form Field. Refer the below GIF for further reference.



### Drag the form field

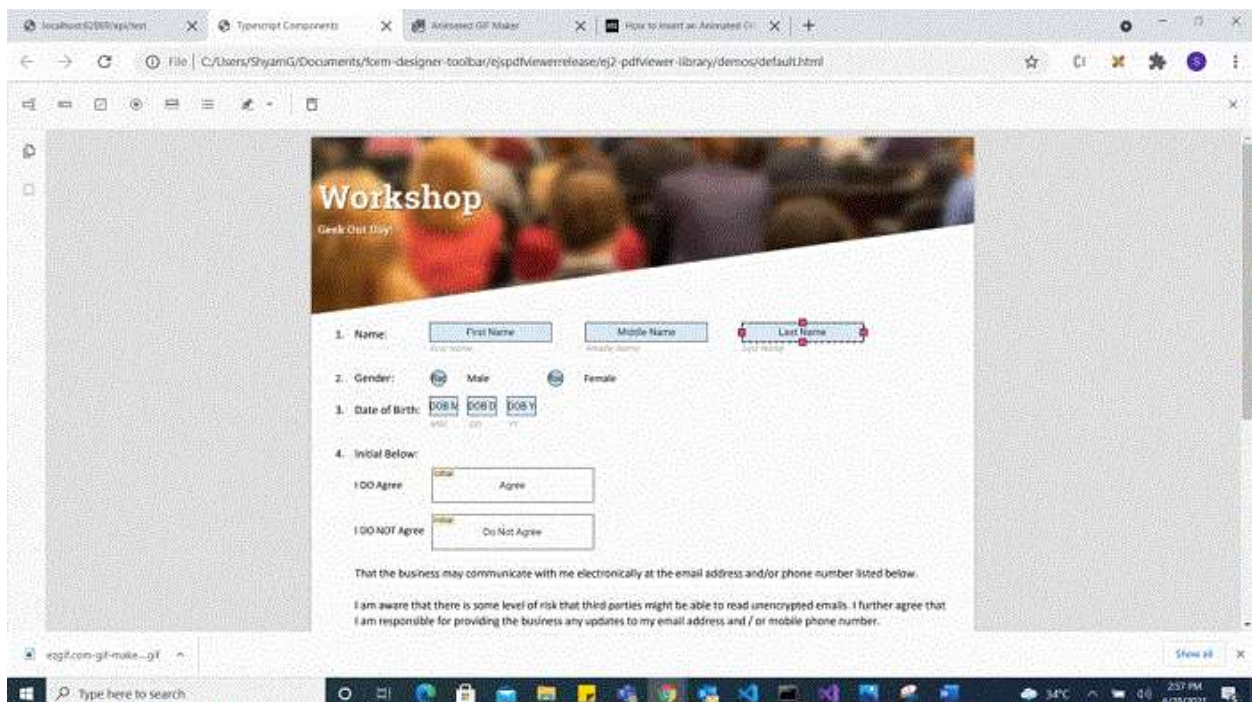
We provide options to drag the Form Field which is currently selected in the PDF document. Refer the below GIF for further reference.





### Resize the form field

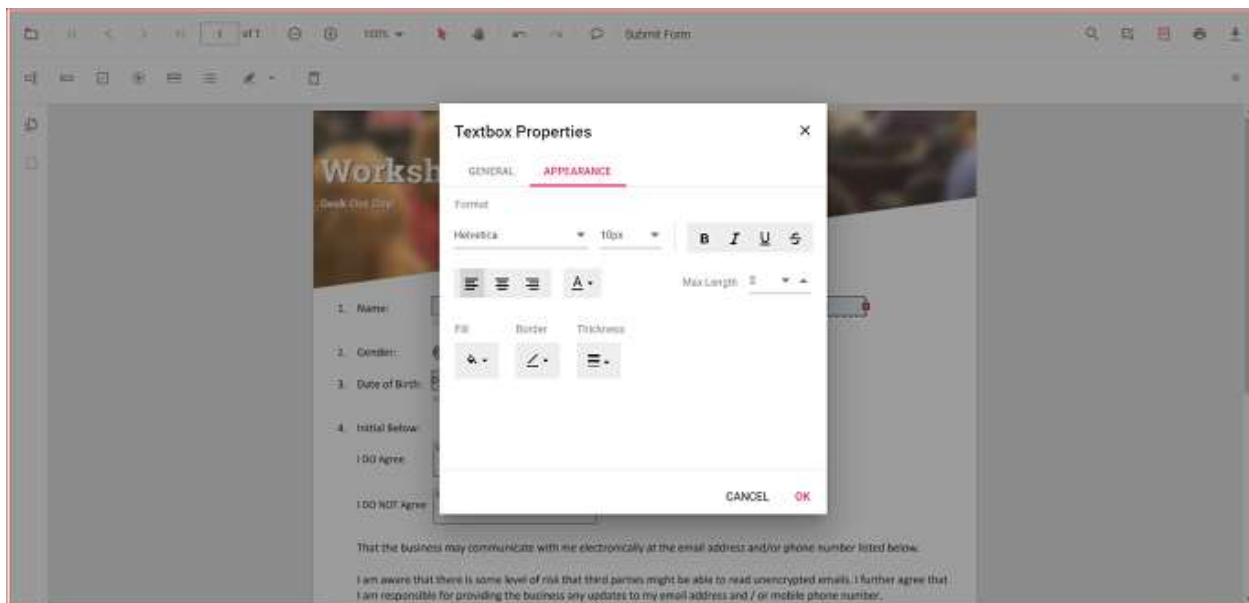
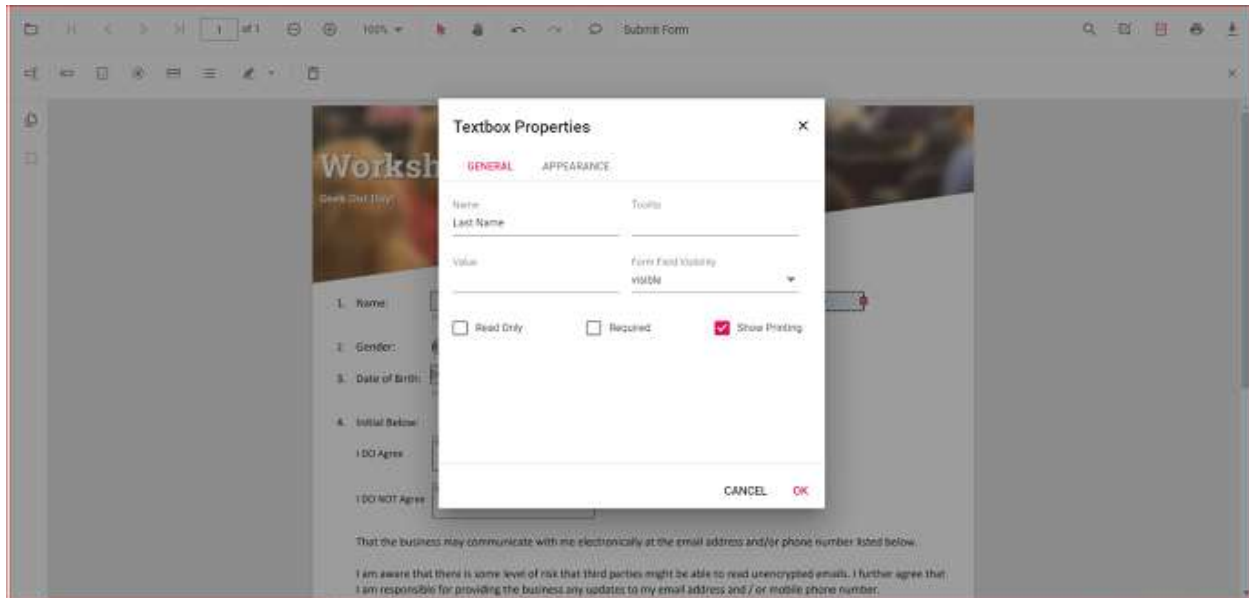
We provide options to resize the Form Field which is currently selected in the PDF document. Refer the below GIF for further reference.

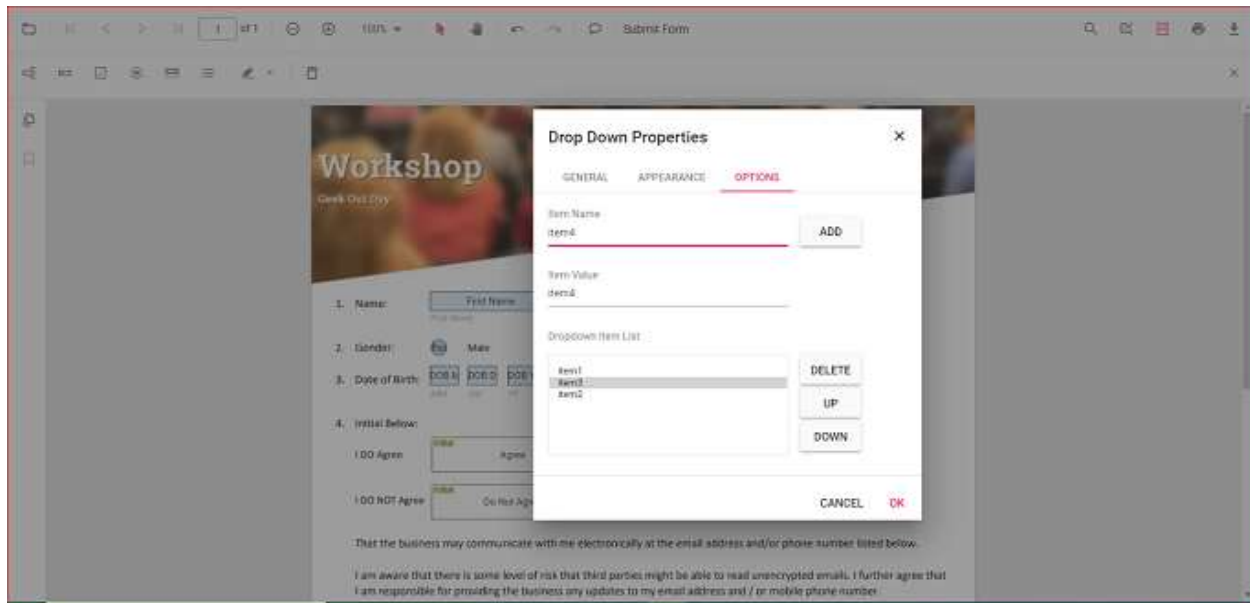


### Edit or Update the form field dynamically

The properties of the Form Fields can be edited using the Form Field Properties window. It can be opened by selecting the Properties option in the context menu that appears on the right by clicking the

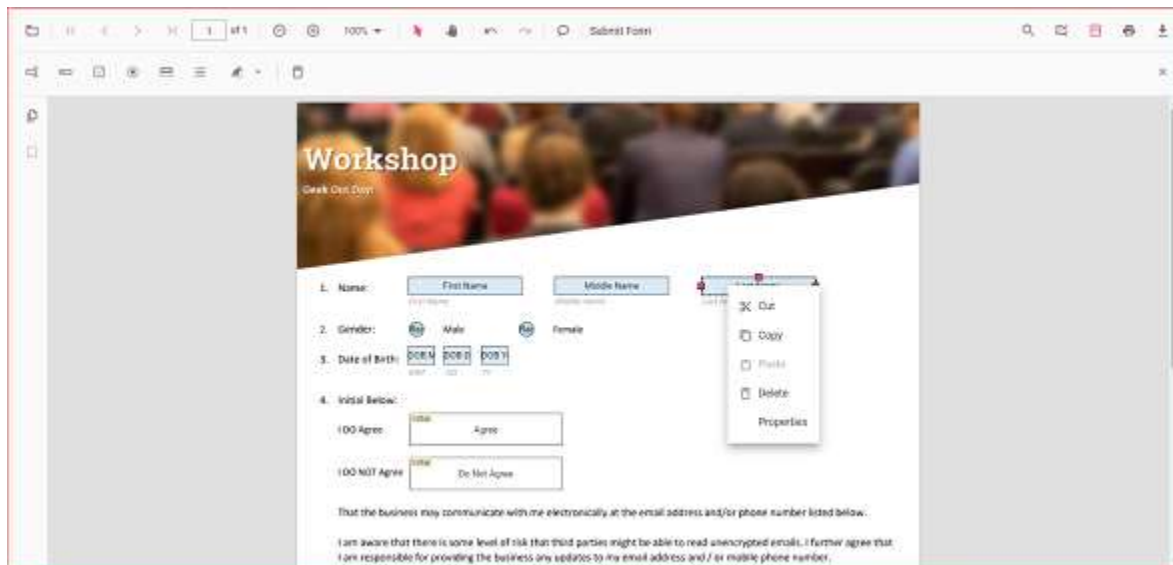
Form Field object. Refer the below image for the properties available to customize the appearance of the Form Field.





### Clipboard operation with form field

The PDF Viewer control supports the clipboard operations such as cut, copy and paste for Form Fields. You can right click on the Form Field object to view the context menu and select to the clipboard options that you would like to perform. Refer the below image for the options in the context menu.



### Undo and Redo

We provided support to undo/redo the Form Field actions that are performed at runtime. Use the following code example to perform undo/redo actions.

```
<button id="undo">Undo</button>
<button id="redo">Redo</button>
```

### STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
 documentPath: "https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
document.getElementById('undo').addEventListener('click', function () {
 pdfviewer.undo();
});
document.getElementById('redo').addEventListener('click', function () {
 pdfviewer.redo();
});
```

### **SERVER-BACKED**

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
 documentPath: "https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
 serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer',
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation,
ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification, ej.pdfviewer.Annotation,
ej.pdfviewer.FormDesigner, ej.pdfviewer.FormFields);
pdfviewer.appendTo('#PdfViewer');
document.getElementById('undo').addEventListener('click', function () {
 pdfviewer.undo();
});
document.getElementById('redo').addEventListener('click', function () {
 pdfviewer.redo();
});
```

### Print in EJ2 JavaScript Pdfviewer control

The PDF Viewer supports printing the loaded PDF file. You can enable/disable the print using the following code snippet.

```
`html
{% raw %}
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 fabric theme -->
<link href="{{CDN_LINK}}ej2-pdfviewer/styles/fabric.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 PDF Viewer's global script -->
<script src="{{CDN_LINK}}dist/ej2.min.js" type="text/javascript"></script>
</head>
```



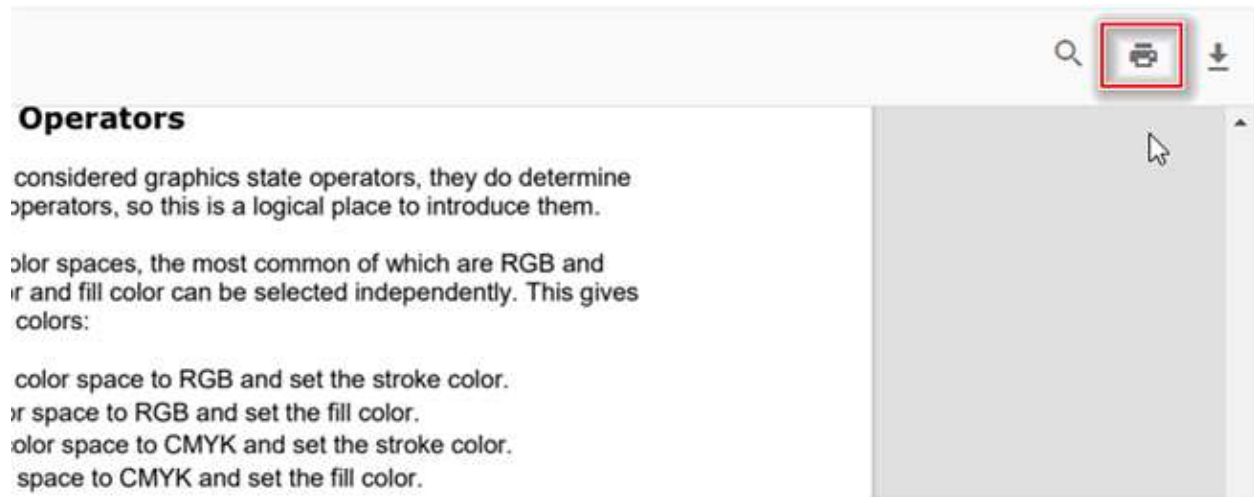
```
<body>
<!--element which is going to render-->
<div id='container'>
<div id='PdfViewer' style="height:500px;width:100%;">
</div>
</div>
</body>
</html>
{% enddraw %}
`
```

### **STANDALONE**

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
enablePrint: true,
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation,ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

### **SERVER-BACKED**

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
enablePrint: true,
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation,ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```



You can invoke print action using the following code snippet.,

,

```
<button id="print">Print</button>
```

,

### STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
 enablePrint: true,
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
document.getElementById('print').addEventListener('click', function () {
 pdfviewer.print.print();
});
```

### SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
 enablePrint: true,
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
 serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
document.getElementById('print').addEventListener('click', function () {
 pdfviewer.print.print();
});
```

See also

- [Toolbar items](#)
- [Feature Modules](#)

### Download in EJ2 JavaScript Pdfviewer control

The PDF Viewer supports downloading the loaded PDF file. You can enable/disable the download using the following code snippet.

```
`html
{% raw %}
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 fabric theme -->
<link href="{{:CDN_LINK}}ej2-pdfviewer/styles/fabric.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 PDF Viewer's global script -->
<script src="{{:CDN_LINK}}dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!--element which is going to render-->
<div id='container'>
<div id='PdfViewer' style="height:500px;width:100%;">
</div>
</div>
</body>
</html>
{% endraw %}
`
```

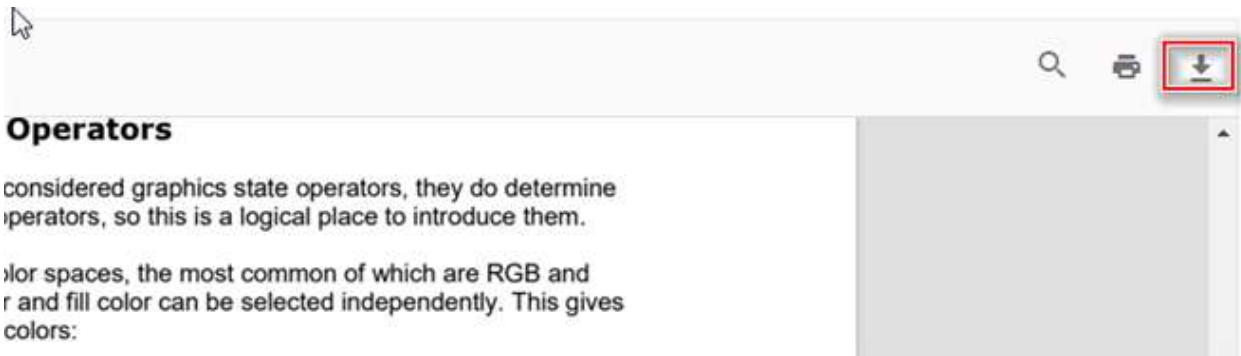
### STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
 enableDownload: true,
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
```

```
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```

### SERVER-BACKED

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
 enableDownload: true,
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
 serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
```



You can invoke download action using following code snippet.,

```
<button id="download">Download</button>
```

### STANDALONE

```
var pdfviewer = new ej.pdfviewer.PdfViewer({
 enableDownload: true,
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
document.getElementById('download').addEventListener('click', function () {
 pdfviewer.download()
});
```

### SERVER-BACKED

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
 enableDownload: true,
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
 serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
document.getElementById('download').addEventListener('click', function () {
 pdfviewer.download()
});

```

### How to get the base64 string while downloading the PDF document

The **downloadEnd** event of the PDF viewer allows you to get the downloaded document as a base64 string.

The following code illustrates how to get the invoke the download action in a button click to get the downloaded document as a base64 string. And load the document from base64 string in another button click.

,

```
<button id="download">Download</button>
```

```
<button id="load">Load</button>
```

,

```
`ts
```

```
var pdfstream;
```

```
document.getElementById('download').addEventListener('click', function () {
```

```
//API to perform download action.
```

```
viewer.download();
```

```
viewer.downloadEnd = function (args) {
```

```
pdfstream = args.downloadDocument;
```

```
//Print the document as a base64 string in the console window.
```

```
console.log(pdfstream);
```

```
};
```

```
});
```

```
document.getElementById('load').addEventListener('click', function () {
```

```
//Load the base64 string in the viewer.
```

```
viewer.load(pdfstream, null);
```

```
});
```

[View sample in GitHub.](#)

See also

- [Toolbar items](#)
- [Feature Modules](#)

### Globalization in EJ2 JavaScript Pdfviewer control

The text contents provided in the PDF Viewer can be localized using the collection of localized strings for different cultures. By default, the PDF Viewer is localized in “**en-US**”.

The following table shows the default text values used in PDF Viewer in 'en-US' culture:

Keywords	Values
---	---
PdfViewer	PDF Viewer
Cancel	Cancel
Download file	Download file
Download	Download
Enter Password	This document is password protected. Please enter a password.
File Corrupted	File corrupted
File Corrupted Content	The file is corrupted and cannot be opened.
Fit Page	Fit page
Fit Width	Fit width
Automatic	Automatic
Go To First Page	Show first page
Invalid Password	Incorrect password. Please try again.
Next Page	Show next page
OK	OK
Open	Open file
Page Number	Current page number
Previous Page	Show previous page
Go To Last Page	Show last page
Zoom	Zoom
Zoom In	Zoom in
Zoom Out	Zoom out
Page Thumbnails	Page thumbnails

|Bookmarks|Bookmarks|  
|Print|Print file  
Password Protected	Password required
Copy	Copy
Text Selection	Text selection tool
Panning	Pan mode
Text Search	Find text
Find in document	Find in document
Match case	Match case
Apply	Apply
GoToPage	Go to page
No matches	Viewer has finished searching the document. No more matches were found
No Text Found	No Text Found
Undo	Undo
Redo	Redo
Annotation	Add or Edit annotations
Highlight	Highlight Text
Underline	Underline Text
Strikethrough	Strikethrough Text
Delete	Delete annotation
Opacity	Opacity
Color edit	Change Color
Opacity edit	Change Opacity
Highlight context	Highlight
Underline context	Underline
Strikethrough context	Strike through
Server error	Web-service is not listening. PDF Viewer depends on web-service for all it's features.
Please start the web service to continue.	
Open text	Open
First text	First Page
Previous text	Previous Page
Next text	Next Page
Last text	Last Page

Zoom in text	Zoom In
Zoom out text	Zoom Out
Selection text	Selection
Pan text	Pan
Print text	Print
Search text	Search
Annotation Edit text	Edit Annotation
General	General
Appearance	Appearance
Options	Options
Textbox Properties	Textbox Properties
Name	Name
Tooltip	Tooltip
Value	Value
Form Field Visibility	Form Field Visibility
Read Only	Read Only
Required	Required
Checked	Checked
Show Printing	Show Printing

The different locale value for the PDF Viewer can be specified using the locale property.

```

`html
{% raw %}
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 fabric theme -->
<link href="{:CDN_LINK}}ej2-pdfviewer/styles/fabric.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 PDF Viewer's global script -->
<script src="{:CDN_LINK}}dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!--element which is going to render-->

```



```

<div id='container'>
<div id='PdfViewer' style="height:500px;width:100%;">
</div>
</div>
</body>
</html>
{% enddraw %}
`

```

### **STANDALONE**

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
 locale: 'ar-AE',
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');

```

### **SERVER-BACKED**

```

var pdfviewer = new ej.pdfviewer.PdfViewer({
 locale: 'ar-AE',
 documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
 serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');

```

You have to map the text content based on locale like following script in sample level.,

```

<script>
ej.base.L10n.load({
 'ar-AE': {
 'PdfViewer': {
 'PdfViewer': 'قوات الدفاع الشعبي المشاهد',
 'Cancel': 'إلغاء',
 'Download file': 'تحميل الملف',

```

'Download': 'تحميل',  
'Enter Password': 'هذا المستند محمي بكلمة مرور. يرجى إدخال كلمة مرور',  
'File Corrupted': 'ملف تالف',  
'File Corrupted Content': 'الملف تالف ولا يمكن فتحه',  
'Fit Page': 'لائق بدنيا الصفحة',  
'Fit Width': 'لائق بدنيا عرض',  
'Automatic': 'تلقائي',  
'Go To First Page': 'عرض الصفحة الأولى',  
'Invalid Password': 'كلمة سر خاطئة. حاول مرة أخرى',  
'Next Page': 'عرض الصفحة التالية',  
'OK': 'حسنًا',  
'Open': 'فتح الملف',  
'Page Number': 'رقم الصفحة الحالية',  
'Previous Page': 'عرض الصفحة السابقة',  
'Go To Last Page': 'عرض الصفحة الأخيرة',  
'Zoom': 'تكبير',  
'Zoom In': 'تكبير في',  
'Zoom Out': 'تكبير خارج',  
'Page Thumbnails': 'مصغرات الصفحة',  
'Bookmarks': 'المرجعية',  
'Print': 'اطبع الملف',  
'Password Protected': 'كلمة المرور مطلوبة',  
'Copy': 'نسخ',  
'Text Selection': 'أداة اختيار النص',  
'Panning': 'وضع عموم',  
'Text Search': 'بحث عن نص',  
'Find in document': 'ابحث في المستند',  
'Match case': 'حالة مباراة',  
'Apply': 'تطبيق',  
'GoToPage': 'انتقل إلى صفحة',  
// tslint:disable-next-line:max-line-length  
'No matches': 'انتهى العارض من البحث في المستند. لم يتم العثور على مزيد من التطابقات',  
'No Text Found': 'لم يتم العثور على نص',

'Undo' : 'فك',  
'Redo' : 'فعل ثانية',  
'Annotation': 'إضافة أو تعديل التعليقات التوضيحية',  
'Highlight': 'تسليط الضوء على النص',  
'Underline': 'تسطير النص',  
'Strikethrough': 'نص يتوسطه خط',  
'Delete': 'حذف التعليق التوضيحي',  
'Opacity': 'غموض',  
'Color edit': 'غير اللون',  
'Opacity edit': 'تغيير التعتيم',  
'Highlight context': 'تسليط الضوء',  
'Underline context': 'أكد',  
'Strikethrough context': 'يتوسطه',  
// tslint:disable-next-line:max-line-length  
'Server error': 'خدمة الانترنت لا يستمع. يعتمد قوات الدفاع الشعبي المشاهد على خدمة الويب لجميع ميزاته. يرجى بدء خدمة',  
'Open text': 'افتح',  
'First text': 'الصفحة الأولى',  
'Previous text': 'الصفحة السابقة',  
'Next text': 'الصفحة التالية',  
'Last text': 'آخر صفحة',  
'Zoom in text': 'تكبير',  
'Zoom out text': 'تصغير',  
'Selection text': 'اختيار',  
'Pan text': 'مقلاة',  
'Print text': 'طباعة',  
'Search text': 'بحث',  
'Annotation Edit text': 'تحرير التعليق التوضيحي',  
'General': 'جنرال لواء',  
'Appearance': 'مظهر خارجي',  
'Options': 'والخيارات',  
'Textbox Properties': 'خصائص مربع النص',  
'Tooltip': 'تلميح',

```
'Value': 'القيمة',
'Form Field Visibility': 'رؤية حقل النموذج',
'Read Only': 'يقرأ فقط',
'Required': 'مطلوب',
'Checked': 'التحقق',
'Show Printing': 'عرض الطباعة'
}
}
});
</script>
`
```

## Server Deployment

### Pdfviewer server docker image overview in EJ2 JavaScript Pdfviewer control

The Syncfusion PDF Viewer control allows you to view, print, form-fill, and annotate PDF files in your web applications. This PDF Viewer control requires a server-side backend Web API service to render PDF contents.

This Docker image is the predefined Docker container of Syncfusion's PDF Viewer backend. You can deploy it quickly to your infrastructure.

PDF Viewer is a commercial product, and it requires a valid license to use it in a production environment ([request license or trial key](#)).

PDF Viewer control is supported in the JavaScript, Angular, React, Vue, ASP.NET Core, ASP.NET MVC, and Blazor platforms.

### Prerequisites

Have [Docker](#) installed in your environment:

- On Windows, install [Docker for Windows](#).
- On macOS, install [Docker for Mac](#).

### How to use this PDF Viewer Docker image

**Step 1:** Pull the pdfviewer-server image from Docker Hub.

```
`console
docker pull syncfusion/pdfviewer-server
`
```

**Step 2:** Create the docker-compose.yml file with the following code in your file system.

```
`yaml
version: '3.4'
services:
```

pdfviewer-server:

image: syncfusion/pdfviewer-server:latest

environment:

[Provide your license key for activation](#)

SYNCFUSIONLICENSEKEY: YOURLICENSEKEY

ports:

- "6001:80"

,

**Step 3:** In a terminal tab, navigate to the directory where you've placed the docker-compose.yml file and execute the following.

```
`console
```

```
docker-compose up
```

,

Also, you can run the Docker container along with the license key using this docker run command.

```
`console
```

```
docker run -d -p 6001:80 -e SYNCFUSIONLICENSEKEY= YOURLICENSEKEY syncfusion/pdfviewer-server:latest
```

,

Now the PDF Viewer server Docker instance runs in the localhost with the provided port number `http://localhost:6001`. Open this link in the browser and navigate to the PDF Viewer Web API control `http://localhost:6001/api/pdfviewer`. It returns the default get method response.

**Step 4:** Append the Docker instance running the URL (`http://localhost:6001/api/pdfviewer`) to the service URL in the client-side PDF Viewer control. For more information about how to get started with PDF Viewer control, refer to this [getting started page](#).

```
`html
```

```
<!DOCTYPE html><html xmlns="http://www.w3.org/1999/xhtml"><head>
```

```
<title>Essential JS 2</title>
```

```
<link href="//cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet">
```

```
<link href="//cdn.syncfusion.com/ej2/ej2-pdfviewer/styles/material.css" rel="stylesheet">
```

```
<link href="//cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet">
```

```
<link href="//cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet">
```

```
<link href="//cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet">
```

```
<link href="//cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css" rel="stylesheet">
```

```
<link href="//cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet">
```

```
<link href="//cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="//cdn.syncfusion.com/ej2/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="//cdn.syncfusion.com/ej2/ej2-drawings/styles/material.css" rel="stylesheet">
<link href="//cdn.syncfusion.com/ej2/ej2-inplace-editor/styles/material.css" rel="stylesheet">
<link href="//cdn.syncfusion.com/ej2/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="//cdn.syncfusion.com/ej2/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<!-- Essential JS 2 PDF Viewer's global script -->
<script src="//cdn.syncfusion.com/ej2/ej2-pdfviewer/dist/global/ej2-pdfviewer.min.js"
type="text/javascript"></script>
<script src="//cdn.syncfusion.com/ej2/dist/ej2.min.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!--element which is going to render-->
<div id="container">
<div id="PdfViewer" style="height:500px;width:100%;">
</div>
</div>
<script>
// Initialize PDF Viewer component.
var pdfviewer = new ej.pdfviewer.PdfViewer({
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: "http://localhost:6001/api/pdfviewer"
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection, ej.pdfviewer.TextSearch,
ej.pdfviewer.Navigation,ej.pdfviewer.Print);
//PDF Viewer control rendering starts
pdfviewer.appendTo('#PdfViewer');
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
}
```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

#### [How to configure the distributed Redis Cache in this Docker image](#)

The PDF Viewer server library internally caches the loaded document instance and you can extend the cache option to a distributed cache environment. Please follow these steps to configure the Azure Cache for a Redis instance to the PDF Viewer server library using a Docker compose file.

**Step 1:** Create the [Azure Cache for the Redis instance](#) and copy the connection string.

**Step 2:** Provide the connection string to the `REDISCACHECONNECTIONSTRING` environment variable in the `pdfviewer-server docker-compose file`. The default cache sliding expiration time is 10 minutes. You can also configure it by setting the value to the `DOCUMENTSLIDINGEXPIRATIONTIME` environment variable.

```
`yaml
```

```
version: '3.4'
```

```
services:
```

```
pdfviewer-server:
```

```
image: syncfusion/pdfviewer-server:latest
```

```
environment:
```

[Provide your license key for activation](#)

```
SYNCFUSIONLICENSEKEY: YOURLICENSEKEY
```

```
REDISCACHECONNECTIONSTRING: YOURREDISCACHECONNECTION_STRING
```

```
DOCUMENTSLIDINGEXPIRATION_TIME: "20"
```

```
ports:
```

- "6001:80"

Refer to these getting started pages to create a PDF Viewer in [Angular](#), [React](#), [Vue](#), [ASP.NET MVC](#), [ASP.NET Core](#), and [Blazor](#).

How to deploy docker image in azure app service for container in EJ2 JavaScript Pdfviewer control

#### [Prerequisites](#)

- Have [Azure account](#) and [Azure CLI](#) setup in your environment.
- Run the following command to open the Azure login page. Sign into your [Microsoft Azure account](#).

az login

,

**Step 1:** Create a resource group.

Create a resource group using the [az group create](#) command.

The following example creates a resource group named pdfviewerresourcegroup in the eastus location.

,

```
az group create --name pdfviewerresourcegroup --location "East US"
```

,

**Step 2:** Create an Azure App Service plan.

Create an App Service plan in the resource group with the [az appservice plan create](#) command.

The following example creates an App Service plan named pdfviewerappservice in the Standard pricing tier (--sku S1) and in a Linux container (--is-linux).

,

```
az appservice plan create --name pdfviewerappservice --resource-group pdfviewerresourcegroup --sku S1 --is-linux
```

,

**Step 3:** Create a Docker Compose app.

Create a multi-container [web app](#) in the pdfviewerappservice App Service plan with the [az webapp create](#) command. The following command creates the web app using the provided Docker compose file. Please look into the section for getting started with Docker compose to create the Docker compose file for the PDF Viewer server and use the created Docker compose file here.

,

```
az webapp create --resource-group pdfviewerappservice --plan pdfviewerappservice --name pdfviewer-server --multicontainer-config-type compose --multicontainer-config-file pdfviewer-server-compose.yml
```

,

**Step 4:** Browse to the app.

Browse to the deployed app at `http://<app_name>.azurewebsites.net`, i.e.

`http://pdfviewerappservice.azurewebsites.net`. Open this link in a browser and navigate to the PDF Viewer Web API control `http://pdfviewerappservice.azurewebsites.net/api/pdfviewer`. It returns the default get method response.

Append the app service running the URL

`http://pdfviewerappservice.azurewebsites.net/api/pdfviewer` to the service URL in the client-side PDF Viewer control. For more information about how to get started with the PDF Viewer control, refer to this [getting started page](#).

For more information about the app container service, please look deeper into the [Microsoft Azure Container Service](#) for a production-ready setup.



## How to deploy docker image in azure kubernetes service in EJ2 JavaScript Pdfviewer control

### Prerequisites

- Have [Azure account](#) and [Azure CLI](#) setup in your environment.
- Run the following command to open the Azure login page. Sign into your [Microsoft Azure account](#).

,

```
az login
```

,

#### Step 1: Create a resource group.

Create a resource group using the [az group create](#) command.

The following example creates a resource group named pdfviewerresourcegroup in the eastus location.

,

```
az group create --name pdfviewerresourcegroup --location "East US"
```

,

#### Step 2: Create AKS cluster.

Use the [az aks create](#) command to create an AKS cluster. The following example creates a cluster named pdfviewercluster with one node.

,

```
az aks create --resource-group pdfviewerresourcegroup --name pdfviewercluster --node-count 1
```

,

#### Step 3: Connect to the cluster.

Install the [kubectl](#) into the workspace using the following command.

,

```
az aks install-cli
```

,

To configure kubectl to connect to your Kubernetes cluster, use the [az aks get-credentials](#) command. This command downloads credentials and configures the Kubernetes CLI to use them.

,

```
az aks get-credentials --resource-group pdfviewerresourcegroup --name pdfviewercluster
```

,

#### Step 4: Create services and deployments.

[Kubernetes Services](#) and [Deployments](#) can be configured in a file. To run the PDF Viewer server, you have to define a Service and a Deployment pdfviewerserver. To do this, create the pdfviewer-server.yaml file in the current directory using the following code.

```
`yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
 app: pdfviewerserver
 name: pdfviewerserver
spec:
 replicas: 1
 selector:
 matchLabels:
 app: pdfviewerserver
 strategy: {}
 template:
 metadata:
 labels:
 app: pdfviewerserver
 spec:
 containers:
 - image: syncfusion/pdfviewerserver:latest

name: pdfviewerserver
ports:
 - containerPort: 80

env:
 - name: SYNCFUSIONLICENSEKEY
 value: "YOURLICENSEKEY"
apiVersion: v1
kind: Service
metadata:
 labels:
 app: pdfviewerserver
```

```
name: pdfviewerserver
```

```
spec:
```

```
ports:
```

- port: 80

```
targetPort: 80
```

```
selector:
```

```
app: pdfviewerserver
```

```
type: LoadBalancer
```

```
,
```

**Step 5:** To create all Services and Deployments needed to run the PDF Viewer server, execute the following.

```
`console
```

```
kubectl create -f ./pdfviewer-server.yaml
```

```
,
```

Run the following command to get the Kubernetes cluster deployed with service details and copy the external IP address of the PDF Viewer service.

```
`console
```

```
kubectl get all
```

```
,
```

Browse the copied external IP address and navigate to the PDF Viewer Web API control `http://<external-ip>/api/pdfviewer`. It returns the default get method response.

**Step 6:** Append the Kubernetes service running the URL `http://<external-ip>/api/pdfviewer` to the service URL in the client-side PDF Viewer control. For more information about how to get started with the PDF Viewer control, refer to this [getting started page](#).

For more details about the Azure Kubernetes service, please look deeper into [Microsoft Azure Kubernetes Service](#) for a production-ready setup.

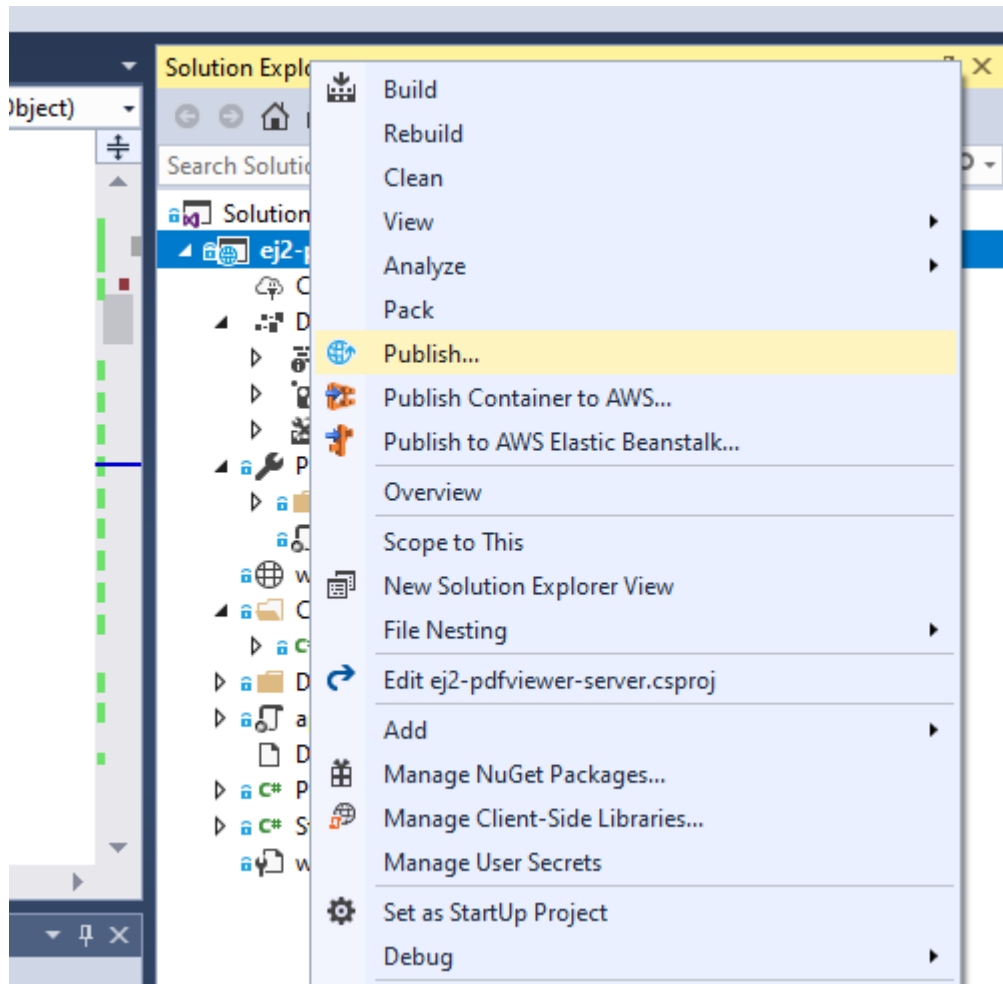
How to deploy pdfviewer server app in azure app service from visual studio in EJ2 JavaScript Pdfviewer control

*Prerequisites*

- Visual Studio 2017 or Visual Studio 2019.
- An [Azure subscription](#).
- Create the [PDF Viewer Web API application](#).
- Make sure you've built the project using the Build > Build Solution menu command before following the deployment steps.

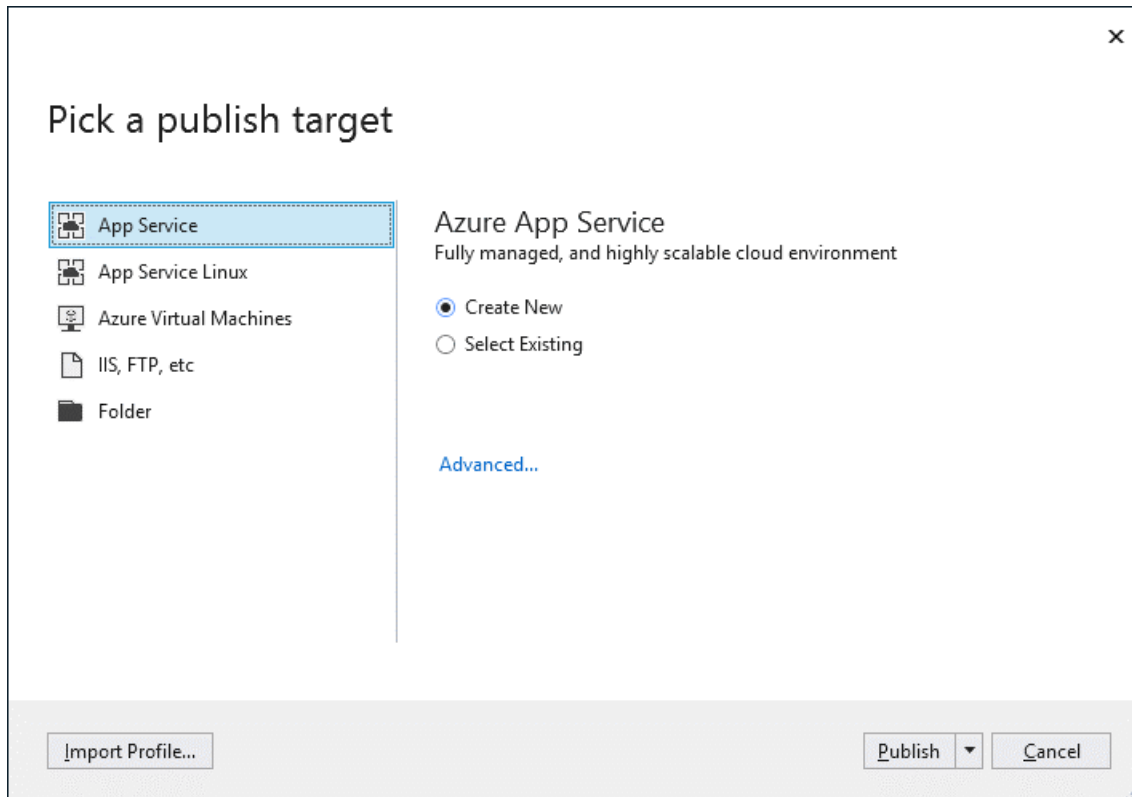
*Publish to Azure App Service*

**Step 1:** In Solution Explorer, right-click the project and choose Publish (or use the Build > Publish menu item).

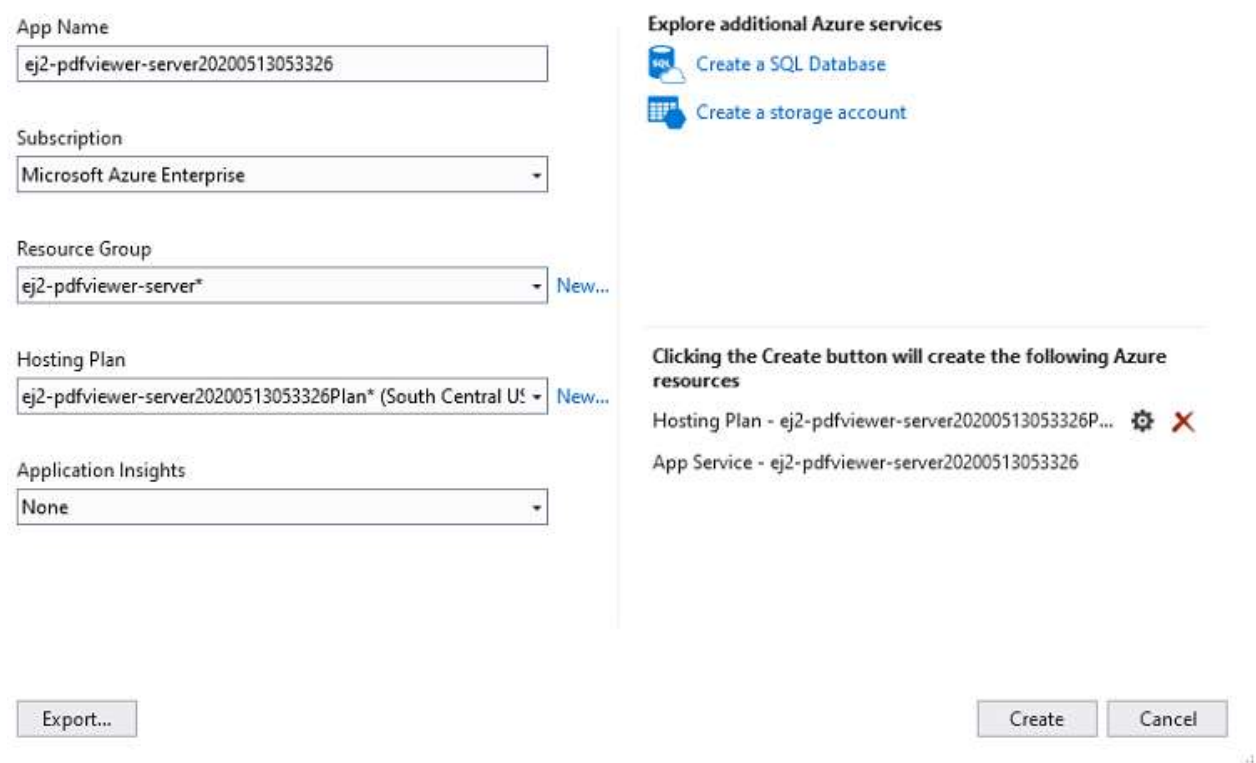


**Step 2:** If you have previously configured any publishing profiles, the Publish pane appears, in which case, select Create new profile.

**Step 3:** In the Pick a publish target dialog box, choose App Service.



**Step 4:** Select Publish. The Create App Service dialog box appears. Sign in with your Azure account, if necessary, then the default app service settings populate the fields.



**Step 5:** Select Create. Visual Studio deploys the app to your Azure App Service, and the web app loads in your browser at `http://<app_name>.azurewebsites.net`. (i.e. `http://ej2-pdfviewer-server20200513053326.azurewebsites.net`).

**Step 6:** Navigate to the PDF Viewer Web API control `http://ej2-pdfviewer-server20200513053326.azurewebsites.net/api/pdfviewer`. It returns the default get method response.

Append the app service running the URL `http://ej2-pdfviewer-server20200513053326.azurewebsites.net/api/pdfviewer` to the service URL in the client-side PDF Viewer control. For more information about how to get started with the PDF Viewer control, refer to this [getting started page](#).

For more information about the app container service, please look deeper into the [Microsoft Azure App Service](#) for a production-ready setup.

## How To

### Create pdfviewer service in EJ2 JavaScript Pdfviewer control

The Essential JavaScript PDF Viewer have server side dependency to get the details from PDF Documents for rendering. This section explains how to create the service for PDF Viewer to perform server-side preprocessing of the PDF document to be rendered in client side.

#### Prerequisites

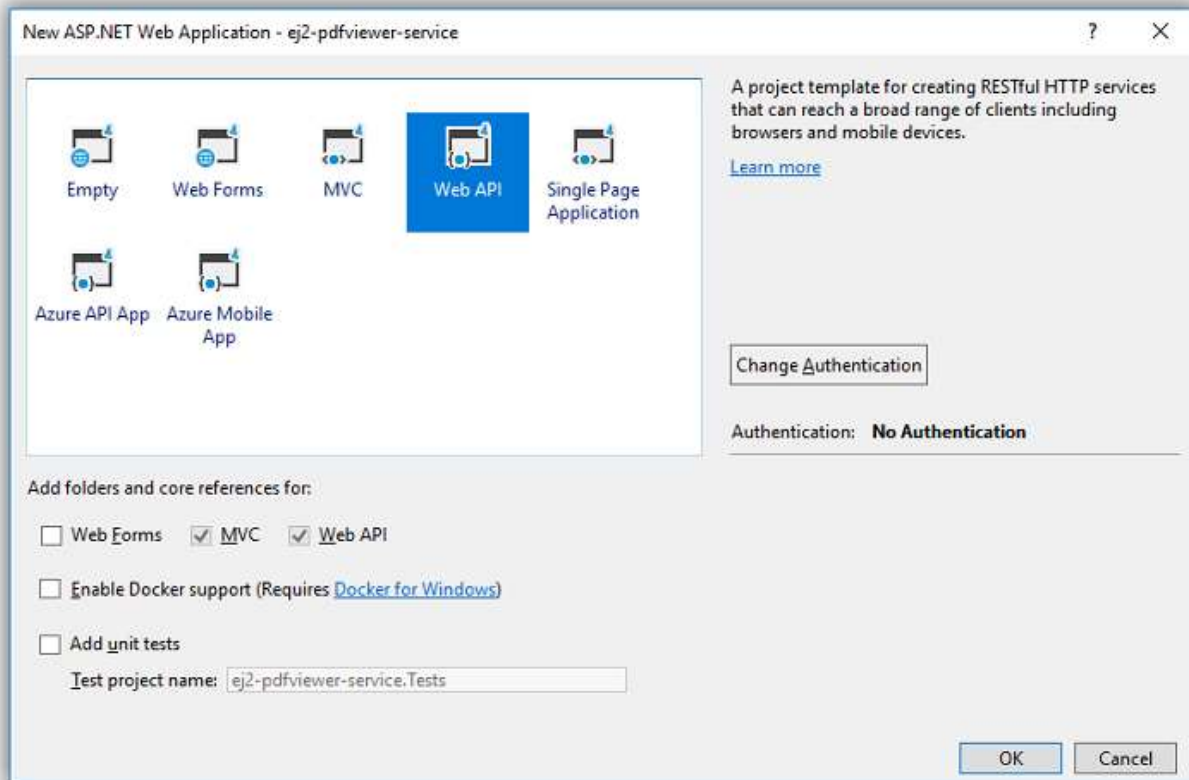
To get started with ASP.NET MVC Web API service, ensure that the following software is installed on the machine.

- .Net Framework 4.5 and above.
- ASP.NET MVC 4 or ASP.NET MVC 5
- Web API
- Visual Studio

#### Setup ASP.NET MVC application with Web API for PDF Viewer service

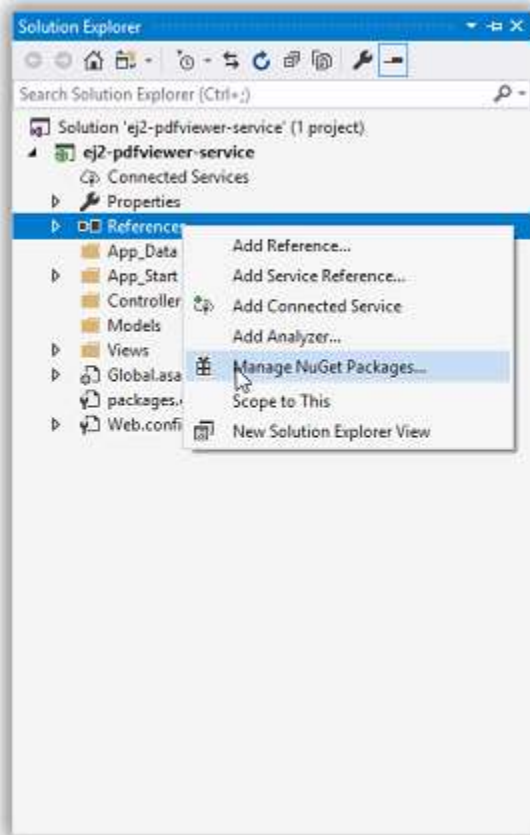
The following steps are used to create PDF Viewer service

**Step 1:** Create an ASP.NET web application with the default template project in Visual Studio 2017.

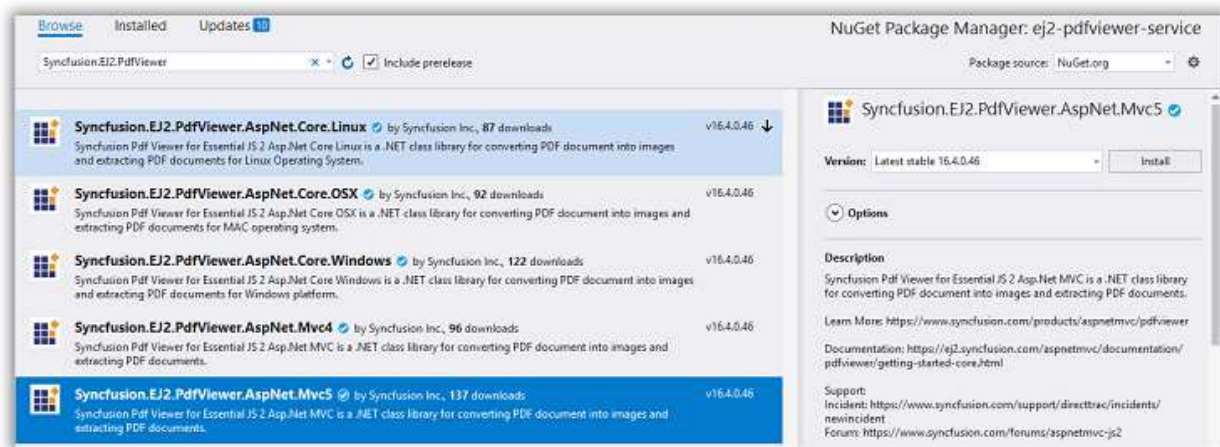


**Step 2:** After creating the project, add the `Syncfusion.EJ2.PdfViewer.AspNet.MVC5` dependency to your project by using 'NuGet Package Manager'.

Open the `NuGet` package manager.

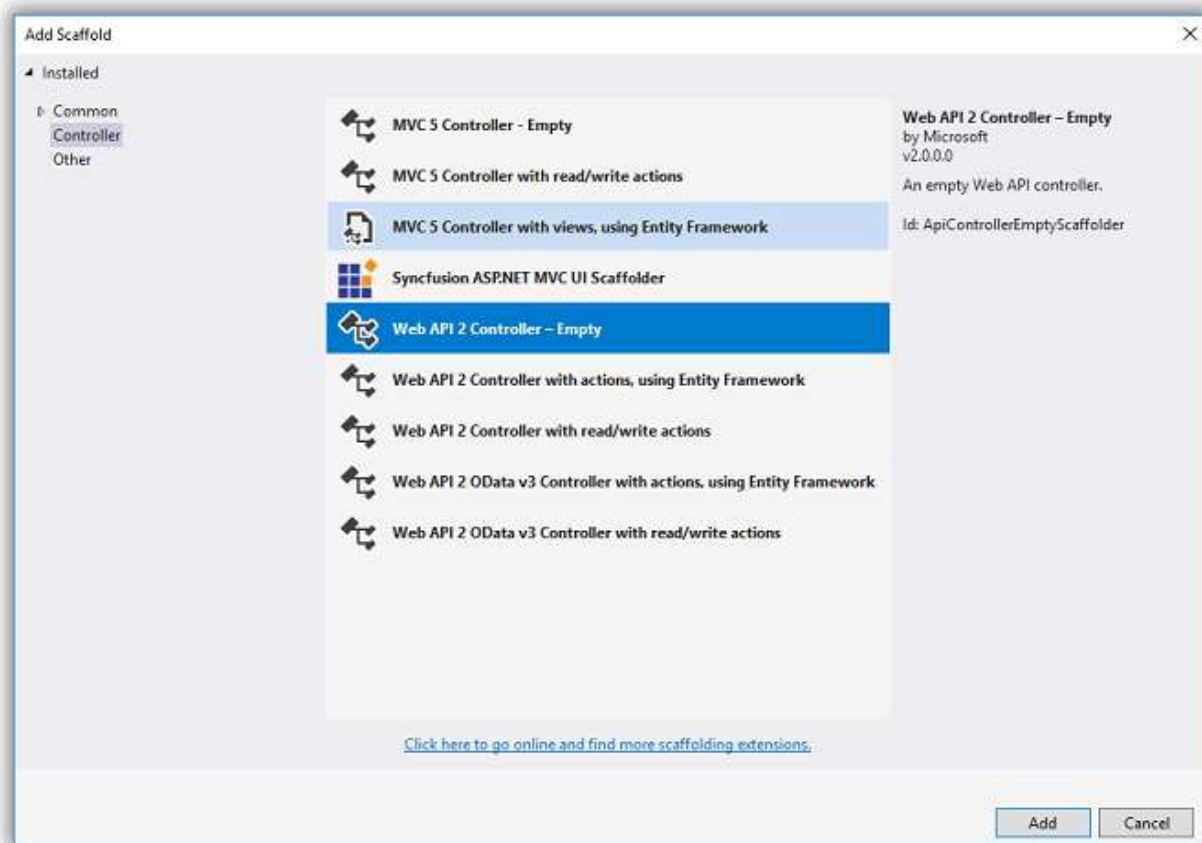


Install the **Syncfusion.EJ2.PdfViewer.AspNet.Mvc5** package to the application.



**Step 3:** Add the Web API 2 Controller to the project and named it as PdfViewerController .





**Step 4:** Add the following code to the PdfViewerController.cs controller.

```
`ts
using Newtonsoft.Json;
using Syncfusion.EJ2.PdfViewer;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web;
using System.Web.Http;
namespace MvcWebService.webapi
{
 public class PdfViewerController : ApiController
 {
```

```
[System.Web.Mvc.HttpPost]
public object Load(Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 MemoryStream stream = new MemoryStream();
 object jsonResult = new object();
 if (jsonObject != null && jsonObject.ContainsKey("document"))
 {
 if (bool.Parse(jsonObject["isFileName"]))
 {
 string documentPath = GetDocumentPath(jsonObject["document"]);
 if (!string.IsNullOrEmpty(documentPath))
 {
 byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
 stream = new MemoryStream(bytes);
 }
 else
 {
 string fileName = jsonObject["document"].Split(new string[] { "://" }, StringSplitOptions.None)[0];
 if (fileName == "http" || fileName == "https")
 {
 WebClient WebClient = new WebClient();
 byte[] pdfDoc = WebClient.DownloadData(jsonObject["document"]);
 stream = new MemoryStream(pdfDoc);
 }
 else
 {
 return (jsonObject["document"] + " is not found");
 }
 }
 }
 else
 {
 return (jsonObject["document"] + " is not found");
 }
 }
}
```

```
byte[] bytes = Convert.FromBase64String(jsonObject["document"]);
stream = new MemoryStream(bytes);
}
}
jsonResult = pdfviewer.Load(stream, jsonObject);
return (JsonConvert.SerializeObject(jsonResult));
}
[System.Web.Mvc.HttpPost]
public object Bookmarks(Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonResult = pdfviewer.GetBookmarks(jsonObject);
 return (jsonResult);
}
[System.Web.Mvc.HttpPost]
public object RenderPdfPages(Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 object jsonResult = pdfviewer.GetPage(jsonObject);
 return (JsonConvert.SerializeObject(jsonResult));
}
[System.Web.Mvc.HttpPost]
public object RenderThumbnaillImages(Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 object result = pdfviewer.GetThumbnaillImages(jsonObject);
 return (JsonConvert.SerializeObject(result));
}
[System.Web.Mvc.HttpPost]
public object RenderPdfTexts(Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 object result = pdfviewer.GetDocumentText(jsonObject);
```

```
return (JsonConvert.SerializeObject(result));
}

[System.Web.Mvc.HttpPost]
public object RenderAnnotationComments(Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 object jsonResult = pdfviewer.GetAnnotationComments(jsonObject);
 return (JsonConvert.SerializeObject(jsonResult));
}

[System.Web.Mvc.HttpPost]
public object Unload(Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 pdfviewer.ClearCache(jsonObject);
 return ("Document cache is cleared");
}

[System.Web.Mvc.HttpPost]
public HttpResponseMessage Download(Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 string documentBase = pdfviewer.GetDocumentAsBase64(jsonObject);
 return (GetPlainText(documentBase));
}

[System.Web.Mvc.HttpPost]
public object PrintImages(Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 object pageImage = pdfviewer.GetPrintImage(jsonObject);
 return (pageImage);
}

private HttpResponseMessage GetPlainText(string pageImage)
{
 var responseText = new HttpResponseMessage(HttpStatusCode.OK);
```

```
responseText.Content = new StringContent(pagelImage, System.Text.Encoding.UTF8, "text/plain");
return responseText;
}

private string GetDocumentPath(string document)
{
 string documentPath = string.Empty;
 if (!System.IO.File.Exists(document))
 {
 var path = HttpContext.Current.Request.PhysicalApplicationPath;
 if (System.IO.File.Exists(path + "/Data/" + document))
 documentPath = path + "/Data/" + document;
 }
 else
 {
 documentPath = document;
 }
 return documentPath;
}

// GET api/values
[System.Web.Mvc.HttpPost]
public IEnumerable<string> Get()
{
 return new string[] { "value1", "value2" };
}

[System.Web.Mvc.HttpPost]
//Post action to export annotations
[System.Web.Mvc.Route("{id}/ExportAnnotations")]
public HttpResponseMessage ExportAnnotations(Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 string jsonResult = pdfviewer.ExportAnnotation(jsonObject);
 return (GetPlainText(jsonResult));
}
```

```
[System.Web.Mvc.HttpPost]
//Post action to import annotations
public object ImportAnnotations(Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 string jsonResult = string.Empty;
 object JsonResult;
 if (jsonObject != null && jsonObject.ContainsKey("fileName"))
 {
 string documentPath = GetDocumentPath(jsonObject["fileName"]);
 if (!string.IsNullOrEmpty(documentPath))
 {
 jsonResult = System.IO.File.ReadAllText(documentPath);
 }
 else
 {
 return (jsonObject["document"] + " is not found");
 }
 }
 else
 {
 string extension = Path.GetExtension(jsonObject["importedData"]);
 if (extension != ".xdf")
 {
 JsonResult = pdfviewer.ImportAnnotation(jsonObject);
 return (GetPlainText((JsonConvert.SerializeObject(JsonResult))));
 }
 else
 {
 string documentPath = GetDocumentPath(jsonObject["importedData"]);
 if (!string.IsNullOrEmpty(documentPath))
 {
 byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
```

```

jsonObject["importedData"] = Convert.ToBase64String(bytes);
JsonResult = pdfviewer.ImportAnnotation(jsonObject);
return (GetPlainText((JsonConvert.SerializeObject(JsonResult))));
}
else
{
return (jsonObject["document"] + " is not found");
}
}
}
return (GetPlainText((JsonResult)));
}
}
}
,

```

**Step 5:** Configure the CORS policy in the `web.config` file.

```

<?xml
<system.webServer>
<httpProtocol>
<customHeaders>
<add name="Access-Control-Allow-Headers" value="accept, maxdataserviceversion, origin, x-requested-
with, dataserviceversion,content-type" />
<add name="Access-Control-Allow-Origin" value="*" />
<add name="Access-Control-Max-Age" value="1728000" />
</customHeaders>
</httpProtocol>
</system.webServer>
,

```

**Step 6:** Set the Global configuration in the `Global.asax` file:

```

,
System.Web.Http.GlobalConfiguration.Configuration.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "api/{controller}/{action}/{id}",

```

```
defaults: new { id = RouteParameter.Optional };
```

```
AppDomain.CurrentDomain.SetData("SQLServerCompactEditionUnderWebHosting", true);
```

```
,
```

Download the sample to create PDF Viewer web service [here](#)

Create pdfviewer service in EJ2 JavaScript Pdfviewer control

The Essential JavaScript PDF Viewer have server side dependency to get the details from PDF Documents for rendering. This section explains how to create the service for PDF Viewer to perform server-side preprocessing of the PDF document to be rendered in client side.

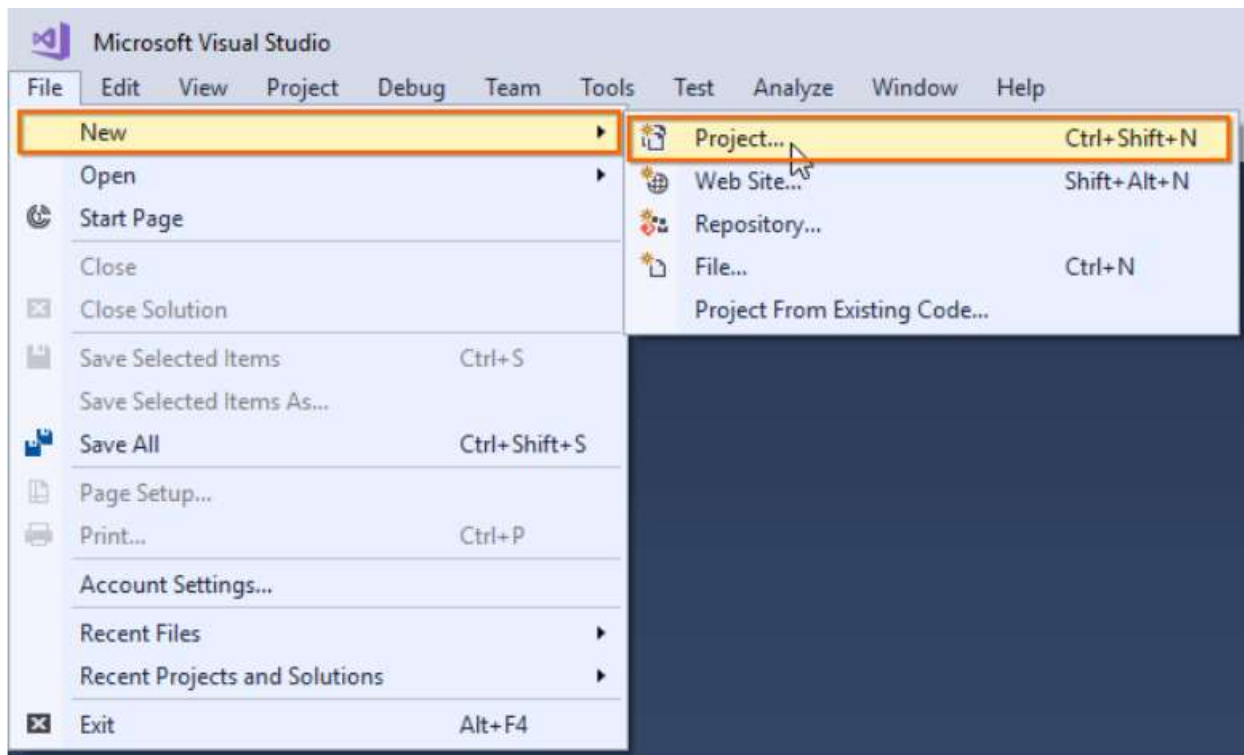
#### *Prerequisites*

To get started with ASP.NET CORE Web API service, refer [System requirements for ASP.NET Core controls](#).

*Setup ASP.NET CORE application with Web API for PDF Viewer service*

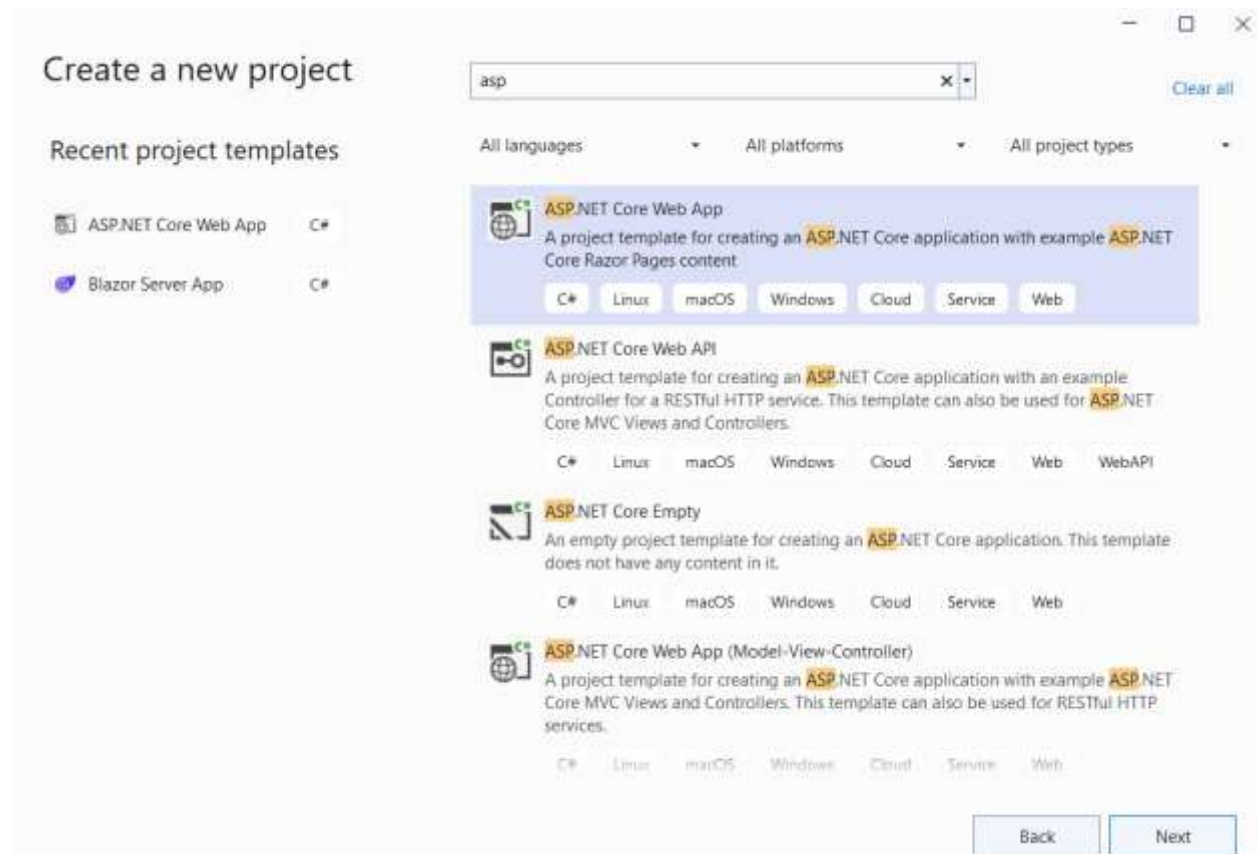
The following steps are used to create PDF Viewer service

**Step 1:** Select File > New > Project, in the Visual Studio menu bar.



**Step 2:** Select ASP.NET Core Web Application and then click **Next**.





**Step 3:** In the Configure your new project dialog, enter Project Name and select **Next**.

Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name

PdfViewerWebService

Location

D:\webservice-aspnetcore\

Solution name ⓘ

PdfViewerWebService

☒ Place solution and project in the same directory

Project will be created in "D:\webservice-aspnetcore\PdfViewerWebService\"

Back Next

**Step 4:** In the Additional information dialog, select .NET 6.0 (Long-term Support) and then select **Create**.

Additional information

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Framework ⓘ  
.NET 6.0 (Long Term Support)

Authentication type ⓘ  
None

☒ Configure for HTTPS ⓘ

☐ Enable Docker ⓘ

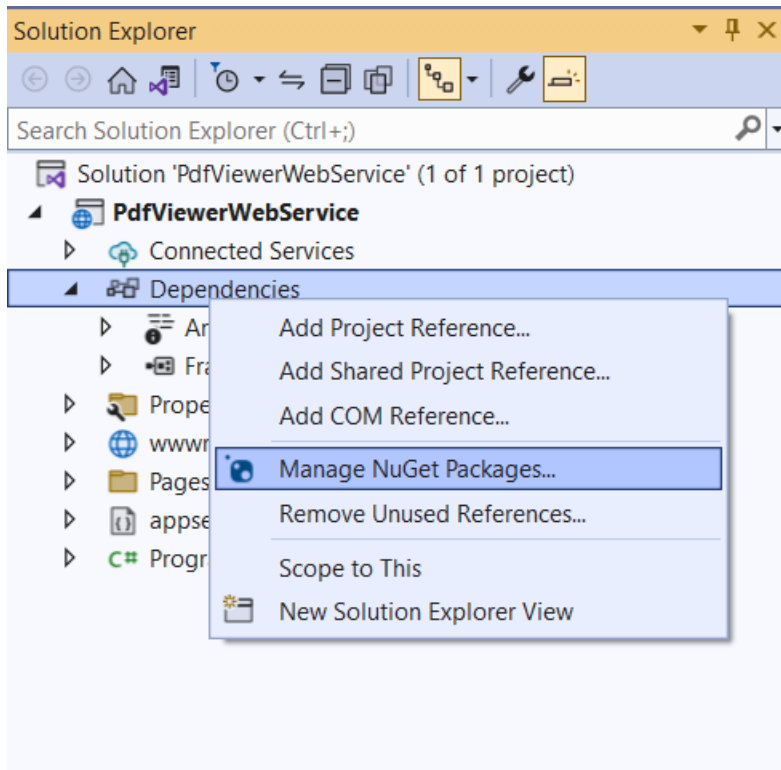
Docker OS ⓘ  
Linux

☐ Do not use top-level statements ⓘ

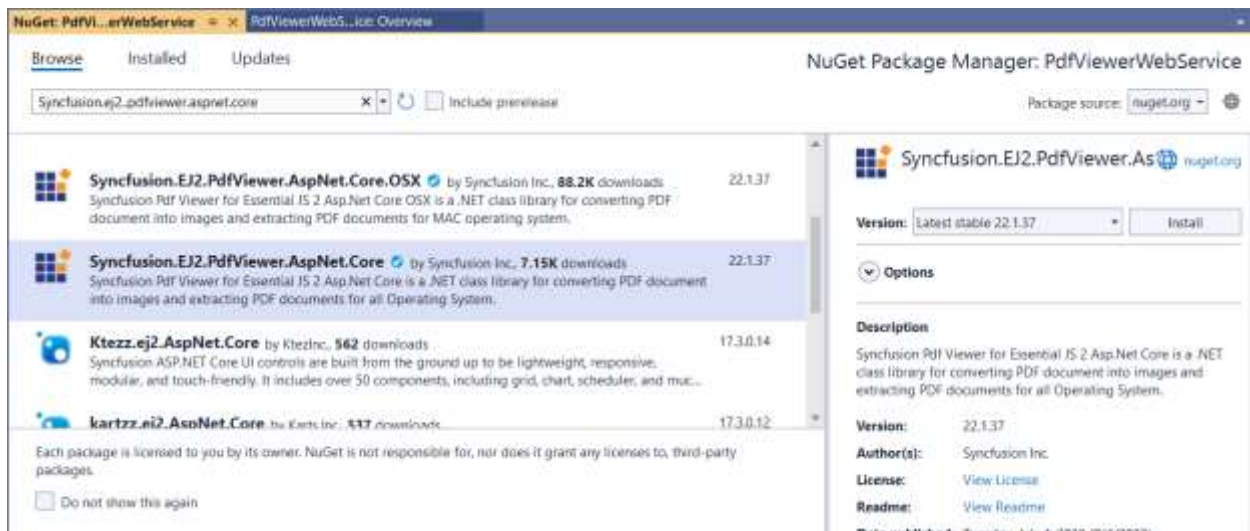
Back Create

**Step 5:** After creating the project, add the [Syncfusion.EJ2.PdfViewer.AspNet.Core](#) dependency to your project by using 'NuGet Package Manager'.

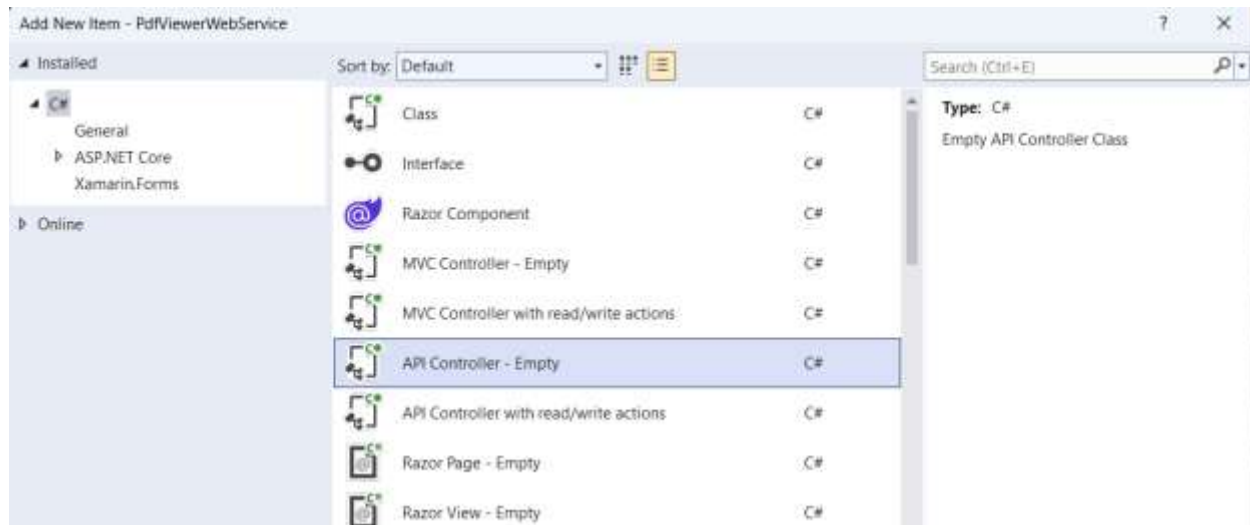
Open the **NuGet** package manager.



Install the **Syncfusion.EJ2.PdfViewer.AspNet.Core** package to the application.



**Step 6:** Add the API Controller to the project and named it as PdfViewerController .



**Step 7:** Add the following code to the PdfViewerController.cs controller.

```
`ts
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Caching.Memory;
using Newtonsoft.Json;
using Syncfusion.EJ2.PdfViewer;
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
namespace PdfViewerWebService
{
 [Route("[controller]")]
 [ApiController]
 public class PdfViewerController : ControllerBase
 {
 private IWebHostEnvironment _hostingEnvironment;
 //Initialize the memory cache object
 public IMemoryCache _cache;
 public PdfViewerController(IWebHostEnvironment hostingEnvironment, IMemoryCache cache)
 {
 _hostingEnvironment = hostingEnvironment;
 }
 }
}
```

```
_cache = cache;
Console.WriteLine("PdfViewerController initialized");
}
[HttpPost("Load")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/Load")]
//Post action for Loading the PDF documents
public IActionResult Load([FromBody] Dictionary<string, string> jsonObject)
{
 Console.WriteLine("Load called");
 //Initialize the PDF viewer object with memory cache object
 PdfRenderer pdfviewer = new PdfRenderer(_cache);
 MemoryStream stream = new MemoryStream();
 object jsonResult = new object();
 if (jsonObject != null && jsonObject.ContainsKey("document"))
 {
 if (bool.Parse(jsonObject["isFileName"]))
 {
 string documentPath = GetDocumentPath(jsonObject["document"]);
 if (!string.IsNullOrEmpty(documentPath))
 {
 byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
 stream = new MemoryStream(bytes);
 }
 else
 {
 string fileName = jsonObject["document"].Split(new string[] { "://" }, StringSplitOptions.None)[0];
 if (fileName == "http" || fileName == "https")
 {
 WebClient WebClient = new WebClient();
 byte[] pdfDoc = WebClient.DownloadData(jsonObject["document"]);
 stream = new MemoryStream(pdfDoc);
 }
 }
 }
 }
}
```

```
else
{
return this.Content(jsonObject["document"] + " is not found");
}
}
}
else
{
byte[] bytes = Convert.FromBase64String(jsonObject["document"]);
stream = new MemoryStream(bytes);
}
}
jsonResult = pdfviewer.Load(stream, jsonObject);
return Content(JsonConvert.SerializeObject(jsonResult));
}
[AcceptVerbs("Post")]
[HttpPost("Bookmarks")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/Bookmarks")]
//Post action for processing the bookmarks from the PDF documents
public IActionResult Bookmarks([FromBody] Dictionary<string, string> jsonObject)
{
//Initialize the PDF Viewer object with memory cache object
PdfRenderer pdfviewer = new PdfRenderer(_cache);
var jsonResult = pdfviewer.GetBookmarks(jsonObject);
return Content(JsonConvert.SerializeObject(jsonResult));
}
[AcceptVerbs("Post")]
[HttpPost("RenderPdfPages")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/RenderPdfPages")]
//Post action for processing the PDF documents
public IActionResult RenderPdfPages([FromBody] Dictionary<string, string> jsonObject)
```

```
{
//Initialize the PDF Viewer object with memory cache object
PdfRenderer pdfviewer = new PdfRenderer(_cache);
object jsonResult = pdfviewer.GetPage(jsonObject);
return Content(JsonConvert.SerializeObject(jsonResult));
}

[AcceptVerbs("Post")]
[HttpPost("RenderPdfTexts")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/RenderPdfTexts")]
//Post action for processing the PDF texts
public IActionResult RenderPdfTexts([FromBody] Dictionary<string, string> jsonObject)
{
//Initialize the PDF Viewer object with memory cache object
PdfRenderer pdfviewer = new PdfRenderer(_cache);
object jsonResult = pdfviewer.GetDocumentText(jsonObject);
return Content(JsonConvert.SerializeObject(jsonResult));
}

[AcceptVerbs("Post")]
[HttpPost("RenderThumbnaillImages")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/RenderThumbnaillImages")]
//Post action for rendering the ThumbnaillImages
public IActionResult RenderThumbnaillImages([FromBody] Dictionary<string, string> jsonObject)
{
//Initialize the PDF Viewer object with memory cache object
PdfRenderer pdfviewer = new PdfRenderer(_cache);
object result = pdfviewer.GetThumbnaillImages(jsonObject);
return Content(JsonConvert.SerializeObject(result));
}

[AcceptVerbs("Post")]
[HttpPost("RenderAnnotationComments")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
```



```
[Route("[controller]/RenderAnnotationComments")]
//Post action for rendering the annotations
public IActionResult RenderAnnotationComments([FromBody] Dictionary<string, string> jsonObject)
{
 //Initialize the PDF Viewer object with memory cache object
 PdfRenderer pdfviewer = new PdfRenderer(_cache);
 object jsonResult = pdfviewer.GetAnnotationComments(jsonObject);
 return Content(JsonConvert.SerializeObject(jsonResult));
}

[AcceptVerbs("Post")]
[HttpPost("ExportAnnotations")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/ExportAnnotations")]
//Post action to export annotations
public IActionResult ExportAnnotations([FromBody] Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer(_cache);
 string jsonResult = pdfviewer.ExportAnnotation(jsonObject);
 return Content(jsonResult);
}

[AcceptVerbs("Post")]
[HttpPost("ImportAnnotations")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/ImportAnnotations")]
//Post action to import annotations
public IActionResult ImportAnnotations([FromBody] Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer(_cache);
 string jsonResult = string.Empty;
 object JsonResult;
 if (jsonObject != null && jsonObject.ContainsKey("fileName"))
 {
 string documentPath = GetDocumentPath(jsonObject["fileName"]);
```

```
if (!string.IsNullOrEmpty(documentPath))
{
 jsonResult = System.IO.File.ReadAllText(documentPath);
}
else
{
 return this.Content(jsonObject["document"] + " is not found");
}
}
else
{
 string extension = Path.GetExtension(jsonObject["importedData"]);
 if (extension != ".xpdf")
 {
 JsonResult = pdfviewer.ImportAnnotation(jsonObject);
 return Content(JsonConvert.SerializeObject(JsonResult));
 }
 else
 {
 string documentPath = GetDocumentPath(jsonObject["importedData"]);
 if (!string.IsNullOrEmpty(documentPath))
 {
 byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
 jsonObject["importedData"] = Convert.ToBase64String(bytes);
 JsonResult = pdfviewer.ImportAnnotation(jsonObject);
 return Content(JsonConvert.SerializeObject(JsonResult));
 }
 else
 {
 return this.Content(jsonObject["document"] + " is not found");
 }
 }
}
```

```
return Content(jsonResult);
}
[AcceptVerbs("Post")]
[HttpPost("ExportFormFields")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/ExportFormFields")]
public IActionResult ExportFormFields([FromBody] Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer(_cache);
 string jsonResult = pdfviewer.ExportFormFields(jsonObject);
 return Content(jsonResult);
}
[AcceptVerbs("Post")]
[HttpPost("ImportFormFields")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/ImportFormFields")]
public IActionResult ImportFormFields([FromBody] Dictionary<string, string> jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer(_cache);
 jsonObject["data"] = GetDocumentPath(jsonObject["data"]);
 object jsonResult = pdfviewer.ImportFormFields(jsonObject);
 return Content(JsonConvert.SerializeObject(jsonResult));
}
[AcceptVerbs("Post")]
[HttpPost("Unload")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/Unload")]
//Post action for unloading and disposing the PDF document resources
public IActionResult Unload([FromBody] Dictionary<string, string> jsonObject)
{
 //Initialize the PDF Viewer object with memory cache object
 PdfRenderer pdfviewer = new PdfRenderer(_cache);
 pdfviewer.ClearCache(jsonObject);
}
```

```
return this.Content("Document cache is cleared");
}

[HttpPost("Download")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/Download")]
//Post action for downloading the PDF documents
public IActionResult Download([FromBody] Dictionary<string, string> jsonObject)
{
 //Initialize the PDF Viewer object with memory cache object
 PdfRenderer pdfviewer = new PdfRenderer(_cache);
 string documentBase = pdfviewer.GetDocumentAsBase64(jsonObject);
 return Content(documentBase);
}

[HttpPost("PrintImages")]
[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]
[Route("[controller]/PrintImages")]
//Post action for printing the PDF documents
public IActionResult PrintImages([FromBody] Dictionary<string, string> jsonObject)
{
 //Initialize the PDF Viewer object with memory cache object
 PdfRenderer pdfviewer = new PdfRenderer(_cache);
 object pageImage = pdfviewer.GetPrintImage(jsonObject);
 return Content(JsonConvert.SerializeObject(pageImage));
}

//Gets the path of the PDF document
private string GetDocumentPath(string document)
{
 string documentPath = string.Empty;
 if (!System.IO.File.Exists(document))
 {
 var path = _hostingEnvironment.ContentRootPath;
 if (System.IO.File.Exists(path + "/Data/" + document))
 documentPath = path + "/Data/" + document;
 }
}
```

```

 }
 else
 {
 documentPath = document;
 }
 Console.WriteLine(documentPath);
 return documentPath;
}
// GET api/values
[HttpGet]
public IEnumerable<string> Get()
{
 return new string[] { "value1", "value2" };
}
// GET api/values/5
[HttpGet("{id}")]
public string Get(int id)
{
 return "value";
}
}
,

```

**Step 8:** Change the launchUrl to pdfviewer (name of the controller) in the lauchSettings.json as follows.

```

`json
{
 "iisSettings": {
 "windowsAuthentication": false,
 "anonymousAuthentication": true,
 "iisExpress": {
 "applicationUrl": "http://localhost:2270",
 "sslPort": 44396
 }
 }
}

```

```

},
"profiles": {
 "PdfViewerWebService": {
 "commandName": "Project",
 "dotnetRunMessages": true,
 "launchBrowser": true,
 "launchUrl": "pdfviewer",
 "applicationUrl": "https://localhost:7255;http://localhost:5262",
 "environmentVariables": {
 "ASPNETCORE_ENVIRONMENT": "Development"
 }
 },
 "IIS Express": {
 "commandName": "IISExpress",
 "launchBrowser": true,
 "launchUrl": "pdfviewer",
 "environmentVariables": {
 "ASPNETCORE_ENVIRONMENT": "Development"
 }
 }
}

```

### Step 9: Configuring CORS policy and add Newtonsoft.Json for JSON format support

- Browser security prevents a webpage from making requests to a different domain than the one that served the webpage. This restriction is called the same-origin policy. Cross Origin Resource Sharing (CORS) allows a server to relax the same-origin policy. Using CORS, a server can explicitly allow some cross-origin. Configure a CORS policy at application Startup.ConfigureServices.
- Prior to ASP.NET Core 3.0, the default JSON formatters implemented using the Newtonsoft.Json package. In ASP.NET Core 3.0 or later, the default JSON formatters are based on System.Text.Json. Support for Newtonsoft.Json based formatters and features is available by installing the Microsoft.AspNetCore.Mvc.NewtonsoftJson NuGet package and configuring it in Startup.ConfigureServices.

```
`cs
```

```
using Microsoft.AspNetCore.ResponseCompression;
```

```
using Newtonsoft.Json.Serialization;
var builder = WebApplication.CreateBuilder(args);
var MyAllowSpecificOrigins = "MyPolicy";
builder.Services.AddControllers().AddNewtonsoftJson(options =>
{
// Use the default property (Pascal) casing
options.SerializerSettings.ContractResolver = new DefaultContractResolver();
});
builder.Services.AddCors(options =>
{
options.AddPolicy(name: MyAllowSpecificOrigins,
builder => {
builder.AllowAnyOrigin()
.AllowAnyMethod()
.AllowAnyHeader();
});
});
builder.Services.AddMemoryCache();
builder.Services.AddEndpointsApiExplorer();
builder.Services.Configure<GzipCompressionProviderOptions>(options => options.Level =
System.IO.Compression.CompressionLevel.Optimal);
builder.Services.AddResponseCompression();
var app = builder.Build();
//Register Syncfusion license
string licenseKey = string.Empty;
Syncfusion.Licensing.SyncfusionLicenseProvider.RegisterLicense(licenseKey);
app.UseHttpsRedirection();
app.UseCors(MyAllowSpecificOrigins);
app.UseAuthorization();
app.UseResponseCompression();
app.MapControllers();
app.Run();
`
```

View the sample in GitHub to [create PDF Viewer web service](#)

### Change author name using annotation settings in EJ2 JavaScript Pdfviewer control

The Essential JavaScript PDF Viewer supports to customize a single property of the annotation settings by exposing an API for the properties common to all the annotations.

**API Name :** annotationSettings

Property Name	Data type & Default Value	Description
---	---	---
author	String("Guest")	specifies the author of the annotation.
minWidth	Number(0)	specifies the minWidth of the annotation.
maxWidth	Number(0)	specifies the maxWidth of the annotation.
minHeight	Number(0)	specifies the minHeight of the annotation.
maxHeight	Number(0)	specifies the maxHeight of the annotation.
isLock	Boolean(false)	specifies the locked action of the annotations. [If set true unable to select the annotations]
isPrint	Boolean(true)	specifies whether the annotations are included or not in Print actions.
isDownload	Boolean(true)	specifies whether the annotations are included or not in Download actions.
Free Text Settings		
allowOnlyTextInput	Boolean(false)	specifies the allow only text action of the free text annotation. [If set true unable to move or resize the annotations]

You can change the author name and the other properties using the annotationSettings API as in the following code sample.

### STANDALONE

```

`ts
import { PdfViewer, Toolbar, Magnification, Navigation, LinkAnnotation,
ThumbnailView, BookmarkView, TextSelection, TextSearch, Print, Annotation,
FormFields } from "../src/index";
PdfViewer.Inject(Toolbar, Magnification, Navigation, LinkAnnotation,
ThumbnailView, BookmarkView, TextSelection, TextSearch, Print, Annotation,
FormFields);
let viewer: PdfViewer = new PdfViewer();
viewer.load('https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf',
null);
viewer.annotationSettings = { author: 'syncfusion', minHeight: 30,
maxHeight: 500, minWidth: 30, maxWidth: 500, isLock: false, isPrint: true,
isDownload: true };
viewer.freeTextSettings = { allowTextOnly : true };
viewer.appendTo("#pdfViewer");
`

```

### SERVER-BACKED

```

`ts

```



```
import { PdfViewer, Toolbar, Magnification, Navigation, LinkAnnotation,
ThumbnailView, BookmarkView, TextSelection, TextSearch, Print, Annotation,
FormFields } from "../src/index";
PdfViewer.Inject(Toolbar, Magnification, Navigation, LinkAnnotation,
ThumbnailView, BookmarkView, TextSelection, TextSearch, Print, Annotation,
FormFields);
let viewer: PdfViewer = new PdfViewer();
viewer.serviceUrl =
"https://services.syncfusion.com/js/production/api/pdfviewer";
viewer.load('https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf',
null);
viewer.annotationSettings = { author: 'syncfusion', minHeight: 30,
maxHeight: 500, minWidth: 30, maxWidth: 500, isLock: false, isPrint: true,
isDownload: true };
viewer.freeTextSettings = { allowTextOnly : true };
viewer.appendTo("#pdfViewer");
`
```

### Customization in EJ2 JavaScript Pdfviewer control

The PDF Viewer provides API for user interactions options provided in it's built-in toolbar. Using this we can create our own User Interface for toolbar actions in application level by hiding the default toolbar. The following steps are used to create the custom toolbar for PDF Viewer,

**Step 1:** Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

**Step 2:** Now, add an HTML div element to render the PDF Viewer with custom toolbar using the following code.

```
`html
{% raw %}
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<link href="{{CDN_LINK}}ej2-base/styles/material.css" rel="stylesheet" />
<link href="{{CDN_LINK}}ej2-pdfviewer/styles/material.css" rel="stylesheet" />
<link href="{{CDN_LINK}}ej2-buttons/styles/material.css" rel="stylesheet" />
<link href="{{CDN_LINK}}ej2-popups/styles/material.css" rel="stylesheet" />
<link href="{{CDN_LINK}}ej2-navigations/styles/material.css" rel="stylesheet" />
<link href="{{CDN_LINK}}ej2-dropdowns/styles/material.css" rel="stylesheet" />
<link href="{{CDN_LINK}}ej2-lists/styles/material.css" rel="stylesheet" />
<link href="{{CDN_LINK}}ej2-inputs/styles/material.css" rel="stylesheet" />
<link href="{{CDN_LINK}}ej2-splitbuttons/styles/material.css" rel="stylesheet" />
<link href="{{CDN_LINK}}ej2-notifications/styles/material.css" rel="stylesheet" />
```

```

<link href="index.css" rel="stylesheet" />
<script src="index.js" type="text/javascript"></script>
</head>
<body>
<div id="container" style="top:0px">...</div>
<div id="topToolbar" style="top:0px"></div>
<div id="magnificationToolbar" ></div>
<div id='viewerParent' style="height:640px; width:100%;">
<div id="pdfViewer" style="height:640px; width:100%;"></div>
</div>
<input type ="file" id="fileUpload" accept=".pdf"
style="display:block;visibility:hidden;width:0;height:0;"/>
<div id='popup'></div>
<div id='textSearchBox'>
<div id='searchContainer'>

<input type="text" id="searchInput" placeholder="Find in document" class="e-input" />

<button id="previousSearch" class="search-button" style="margin-left:5px"></button>
<button id="nextSearch" class="search-button"></button>
</div>
<div id="matchCaseContainer" style="margin-top:8px">
<input id="matchCase" type="checkbox" />
</div>
</div>
</body>
</html>
{% enddraw %}
`

```

**Step 3:** Hide the default toolbar of PDF Viewer using below code snippet,

#### **STANDALONE**

```

` javascript
var pdfviewer = new ej.pdfviewer.PdfViewer({

```

```
enableToolbar: false,
enableThumbnail: false,
documentPath: "https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf"
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
````
```

SERVER-BACKED

```
````javascript
var pdfviewer = new ej.pdfviewer.PdfViewer({
enableToolbar: false,
enableThumbnail: false,
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
pdfviewer.appendTo('#PdfViewer');
````
```

Step 4: Add EJ2 Toolbar for perform primary actions like Open, Previous page, Next page, Go to page, Print and Download using the following code snippet,

```
`javascript
var toolbarObj = new ej.navigations.Toolbar({
items: [
{ prefixIcon: 'e-pv-open-document', tooltipText: 'Open', id: 'openButton', click: openClicked },
{ prefixIcon: 'e-pv-bookmark-icon', tooltipText: 'Bookmark', id: 'bookmarkButton', click:
bookmarkClicked },
{ prefixIcon: 'e-pv-previous-page-navigation-icon', id: 'previousPage', tooltipText: 'Previous Page', align:
'Center', click: previousClicked.bind(this) },
{ prefixIcon: 'e-pv-next-page-navigation-icon', id: 'nextPage', tooltipText: 'Next Page', align: 'Center',
click: nextClicked.bind(this) },
{ template: inputTemplate, tooltipText: 'Page Number', align: 'Center' },
{ template: ele, tooltipText: 'Page Number', align: 'Center' },
{ prefixIcon: 'e-pv-search-icon', tooltipText: 'Text Search', align: 'Right', click: searchClicked.bind(this) },
{ prefixIcon: 'e-pv-print-document-icon', tooltipText: 'Print', align: 'Right', click: printClicked.bind(this) },
{ prefixIcon: 'e-pv-download-document-icon', tooltipText: 'Download', align: 'Right', click:
downloadClicked.bind(this) }
]
});
toolbarObj.appendTo('#topToolbar');
```

```

var magnificationToolbar = new ej.navigations.Toolbar({
items: [
{ prefixIcon: 'e-pv-fit-page-icon', id: 'fitPage', tooltipText: 'Fit to page', click: pageFitClicked.bind(this) },
{ prefixIcon: 'e-pv-zoom-in-icon', id: 'zoomIn', tooltipText: 'Zoom in', click: zoomInClicked.bind(this) },
{ prefixIcon: 'e-pv-zoom-out-icon', id: 'zoomOut', tooltipText: 'Zoom out', click:
zoomOutClicked.bind(this) },
]
});
magnificationToolbar.appendTo('#magnificationToolbar');

```

Step 5: Add the following style to achieve the custom toolbar styling,

```

<style>
magnificationToolbar {
background: transparent;
height: auto;
min-height: 56px;
width: 200px;
border: none;
position: absolute;
z-index: 1001;
top: calc(100% - 110px);
left: calc(100% - 120px);
transform: rotate(90deg);
}
div#magnificationToolbar.e-toolbar .e-toolbar-items {
background: transparent;
}
magnificationToolbar.e-toolbar .e-tbar-btn {
border-radius: 50%;
min-height: 30px;
min-width: 30px;
border: 1px solid #c8c8c8;
transform: rotate(90deg);
}

```

```
}  
topToolbar {  
  top: 0px;  
  z-index: 1001;  
}  
.e-bookmark-popup {  
  height: 300px;  
  max-width: 300px;  
}  
.e-text-search-popup {  
  height: 104px;  
  max-width: 367px;  
}  
.e-text-search-popup .e-footer-content button.e-btn, .e-bookmark-popup .e-footer-content button.e-  
btn {  
  padding: 0;  
  border: 0;  
}  
.e-custom-search-input {  
  width: 234px;  
}  
.e-text-search-popup .e-footer-content, .e-bookmark-popup .e-footer-content {  
  padding: 0;  
  height: 0;  
}  
.search-button, .search-button:disabled, .search-button:focus, .search-button:hover {  
  background: transparent;  
  box-shadow: none;  
  border: 0px;  
}  
popup .e-dlg-content {  
  padding-left: 0;  
  padding-bottom: 0;  
}
```

```
.e-pv-bookmarks {  
min-width: 234px;  
}  
.e-pv-current-page-number {  
width: 46px;  
height: 28px;  
text-align: center;  
}  
.material .e-pv-current-page-number {  
border-width: 1px;  
}  
.e-icons {  
font-family: "e-icons";  
font-style: normal;  
font-variant: normal;  
font-weight: normal;  
line-height: 1;  
text-transform: none;  
}  
.e-pv-icon::before {  
font-family: 'e-icons';  
}  
.e-pv-icon-search::before {  
font-family: 'e-icons';  
font-size: 12px;  
}  
.e-pv-download-document-icon::before {  
content: '\e914';  
}  
.e-pv-print-document-icon::before {  
content: '\e917';  
}  
.e-pv-previous-page-navigation-icon::before {
```

```
content: '\e910';
}
.e-pv-next-page-navigation-icon::before {
content: '\e911';
}
.e-pv-zoom-out-icon::before {
content: '\e912';
}
.e-pv-zoom-in-icon::before {
content: '\e91d';
}
.e-pv-fit-page-icon::before {
content: '\e91b';
}
.e-pv-bookmark-icon::before {
content: '\e915';
}
.e-pv-text-search-icon::before {
content: '\e916';
}
.e-pv-search-icon::before {
content: '\e916';
font-family: 'e-icons';
}
.e-pv-previous-search::before {
content: '\e918';
font-family: 'e-icons';
}
.e-pv-next-search-btn::before {
content: '\e919';
font-family: 'e-icons';
}
.e-pv-open-document::before {
```

```
content: '\e91c';
font-family: 'e-icons';
}
.e-pv-search-close {
content: '\e91a';
font-family: 'e-icons';
}
@font-face {
font-family: "e-icons";
font-style: normal;
font-weight: normal;
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgT1MvMj8wS0QAAAEoAAAAVmNtYXDSenLMAAABuAAAAFZnbHlmok0Nt
wAAAjAAAAPkaGVhZBN3pEcAAADQAAAAANmhoZWEHrwNhAAAArAAAACRobXR4NsgAAAAAYAAAAA4b
G9jYQdkBmQAAAIQAAAAHm1heHABHAAwAAABCAAAACBuYW1lD0oZXgAABhQAAALBcG9zdFG4mE4AA
AjYAAAAyAABAAADUv9qAFoEAAAA/+gEAAAABAAAAAAAAAAAAAAAAAADgABAAAAAQAAxsly1F8PPPU
ACwPoAAAAANgsr7EAAAAA2CyvsQAAAAEAAQAAAAACAACAAAAAAAAAAAAEAAAAOACQABAAAAAAAag
AAAAoACgAAAP8AAAAAQAAPqAZAABQAAAanoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA6RDpHQNS/2oAWgQAAJYAAAABAAAAAAAAABAAAAAPoAAAD6
AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAAAACAAAAAwAA
ABQAAwABAAAAFAAEAEIAAAAGAAQAAQAC6RLpHf//AADpEOkU//8AAAAAAAEABgAKAAAAAQACAAMA
BQAGAAcACAIAAoACwAMAA0ABAAAAAAAAAAUACoAZACkAL4A7gEuAVwBcAGEAZ4ByAHyAAAAAQAA
AAD6gMuAAUAAAKBBwkBJwIAAet0/on+iXQDL/4VcwF3/olzAAEAAAAAA+oDLgAFAAAATCQEXCQGJAXcB
d3T+ff4VAY/+iQF3c/4VAesAAAAAAwAAAAEAAQAAAMADwAbAAABITUhBQ4BBy4BJz4BNx4BBRYAFzYA
NyYAJwYAAQACAP4AAoAE2aOj2QQE2aOj2fyEBgEh2dkBIQYG/t/Z2f7fAcCAQKPZBATZo6PZBATZo9n+3wY
GASHZ2QEhBgb+3wAAAAADAAAAAQABAAACwAXACMAAAEjFTMVMzUzNSM1lwEOAQcuASc+ATceAQ
UWABc2ADcmACcGAHAwMCAwMCAACAE2aOj2QQE2aOj2fyEBgEh2dkBIQYG/t/Z2f7fAkCAwMCAwP8A
o9kEBNmjo9kEBNmj2f7fBgYBldnZASEGBv7fAIAIAAAAAAwAEAAADAAoAADEhNSEBIQkBIREhAwD9AAEA/
wABgAGA/wD/AIACAP6AAYABgAACAAAAAANABAAADgAaAAABMh4CFREIBRE0Nz4BMycGFREIBRE0JiMh
lgKdCwwHBf7g/uAJBAwKdC8BoAGgX0T+BkQDgAYGCwr9YHZ2AqAOCQQGUS9D/KGrqwNfRIsAAAAACAA
AAP/BAAACwAJAAABDgEHLgEnPgE3HgEFHgEXMjY/ARcVATcBlYc3PgE1LgEnDgECgAOQbW2QAwOQbW
2Q/YME2aNGfDIDJAEYf78MyMCKi4E2aOj2QKAbZADA5BtbZADA5Bto9kELioDJDP+/GEBBCQDMnxGo9k
EBNkAAAAQAAAAABAAEAAADAACAFQAZAAABFSE1JRUjNSERMxUhNTMRLgEnIQ4BNyE1IQLA/oACQID9A
MACgMABSDf9ADdlvwKA/YABwMDAwICA/sDAwAFAN0gBAUmKwAAAAQAAAAACQAQAAAUABEBNwk
BJwHsU/6HAXpSAmD+YGIBPgE+YgAAAAEAAAAAAkAEAAFAAAARCEXCEBEv6HUwHs/hMDnv7C/sJiAa
ABoAABAAAAAAKABAAACwAAERcHFzcXNyc3JwcN9fVM9PVL9PRL9fQDtfX0TPX1TPT0TPT0AAAAABAAAA
AD8APwAAUACwARABcAAcEzNTM1IQUzFTMRISUhNSM1lwUjFSElWJ2fvz+hv2K/H7+hgJ2AXr8fv6G/AF6
fvx+fvwBevx+/Px+AXoAAAAAAGAAAAEAAQAAAMAFgAAAREhEScGFREUFhchPgE1ETQmlyEnIQYDgP0AY
h48LQMULTw8Lf5pa/7ULQMA/gACAN8eLf1YLTwDAzwtAigvPYACAAAAAASAN4AAQAAAAAABAAAA
AAQAAAAAAQAUAEEAAQAAAAAAGAHABUAAQAAAAAAAwAUABwAAQAAAAABAAUADAAAQAAAAA
ABQALAEQAAQAAAAABgAUAE8AAQAAAAAACgAsAGMAAQAAAAACwASAI8AAwABBakAAAAACAKEAA
wABBakAAQAoAKMAAwABBakAAgAOAMsAAwABBakAAwAoANKAAwABBakABAAoAQEAawABBakABQ
AWASKAAwABBakABgAoAT8AAwABBakACgBYAWcAAwABBakACwAkAb8gY3VzdG9tLXRvb2xiYXJBMtKw
```



```

OF1SZWd1bGFyY3VzdG9tLXRvb2xiYXJbMTkwOF1jdXN0b20tdG9vbGJhclsxOTA4XVZlcnNpb24gMS4wY3V
zdG9tLXRvb2xiYXJbMTkwOF1Gb250IGdlbmVyYXRIZCB1c2luZyBTW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3
dy5zeW5jZnVzaW9uLmNvbQAgaGMAdQBzAHQAbwBtAC0AdABvAG8AbABiAGEAcgBbADEAOQAwwADgA
XQBSAGUAZwB1AGwAYQByAGMAdQBzAHQAbwBtAC0AdABvAG8AbABiAGEAcgBbADEAOQAwwADgAXQB
jAHUAcwB0AG8AbQAAtAHQAbwBvAGwAYgBhAHIAWwAxADKAMAA4AF0AVgBIAHIAcwBpAG8AbgAgADE
ALgAwAGMAdQBzAHQAbwBtAC0AdABvAG8AbABiAGEAcgBbADEAOQAwwADgAXQBGAG8AbgB0ACAAZw
BIAG4AZQByAGEAdABIAGQAIAB1AHMAaQBuaGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBIAHQa
cgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAA
gAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAOAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAE
NAQ4BDwAIVG9wLWljB24LZG93bi1hcnJvdzlkUFZfWm9vbW91dAlQVl9ab29taW4LUFZfRG93bmxvYWQL
UFZfQm9va21hcmsJUFZfU2VhcmNoCFBWX1ByaW50C1BWX1ByZXZpb3VzB1BWX05leHQlUFZfQ2xvc2U
MUFZfRml0VG9yQYWdlB1BWX09wZW4AAA==) format('truetype');
}
</style>
`

```

The icons are embedded in the font file used in above code snippet.

Step 6: Add the following scripts for performing user interaction in PDF Viewer in code behind

STANDALONE

```

` `` javascript
var inputTemplate = '<div class=""><input type="text" class="e-input-group
e-pv-current-page-number" id="currentPage" /></div>';
var totalPageNum = '<div class=""><span class="e-pv-total-page-number"
id="totalPage">of 0</span></div>';
var isBookmarkOpen = false;
var isBookmarkClick = false;
var isTextSearchBoxOpen = false;
var bookmarkPopup;
var textSearchPopup;
var toolbarObj;
var viewer;
var currentPageBox;
var searchInput;
var searchButton;
var openDocument;
var matchCase = false;
var fileInputElement;
var filename;
function previousClicked(args) {
hidePopups();
viewer.navigation.goToPreviousPage();
}
function hidePopups() {
isBookmarkOpen = false;
isTextSearchBoxOpen = false;
bookmarkPopup.hide();
textSearchPopup.hide();
}
function bookmarkClicked() {
textSearchPopup.hide();
}

```

```
if (!isBookmarkOpen) {
var bookmarkDetails = viewer.bookmark.getBookmarks();
if (bookmarkDetails.bookmarks) {
var bookmarks = bookmarkDetails.bookmarks.bookMark;
var treeObj = new ej.navigations.TreeView({
fields: {
dataSource: bookmarks,
id: 'Id',
parentID: 'Pid',
text: 'Title',
hasChildren: 'HasChild',
}, nodeSelected: nodeClick
});
treeObj.appendTo('#bookmarkview');
bookmarkPopup.show();
isBookmarkOpen = true;
isBookmarkClick = true;
}
else {
toolbarObj.enableItems(document.getElementById('bookmarkButton'), false);
isBookmarkOpen = false;
}
}
else {
if (!isBookmarkClick) {
bookmarkPopup.show();
isBookmarkClick = true;
} else {
bookmarkPopup.hide();
isBookmarkClick = false;
}
}
}
function nextClicked(args) {
hidePopups();
viewer.navigation.goToNextPage();
}
function searchClicked(args) {
bookmarkPopup.hide();
if (!isTextSearchBoxOpen) {
textSearchPopup.show();
}
else {
viewer.textSearch.cancelTextSearch();
textSearchPopup.hide();
}
isTextSearchBoxOpen = !isTextSearchBoxOpen;
}
function printClicked(args) {
hidePopups();
viewer.print.print();
}
function downloadClicked(args) {
hidePopups();
viewer.download();
}
function pageFitClicked(args) {
```

```

hidePopups();
viewer.magnification.fitToPage();
updateZoomButtons();
toolbarObj.enableItems(document.getElementById('fitPage'), false);
}
function zoomInClicked(args) {
hidePopups();
viewer.magnification.zoomIn();
updateZoomButtons();
}
function zoomOutClicked(args) {
hidePopups();
viewer.magnification.zoomOut();
updateZoomButtons();
}
function readFile(args) {
// tslint:disable-next-line
var uploadedFiles = args.target.files;
if (args.target.files[0] !== null) {
var uploadedFile = uploadedFiles[0];
filename = uploadedFiles[0].name;
if (uploadedFile) {
var reader = new FileReader();
reader.readAsDataURL(uploadedFile);
// tslint:disable-next-line
reader.onload = function (e) {
var uploadedFileUrl = e.currentTarget.result;
viewer.load(uploadedFileUrl, null);
currentPageBox.value = '1';
document.getElementById("bookmarkview").innerHTML = "";
isBookmarkOpen = false;
viewer.fileName = filename;
};
}
}
function openClicked() {
document.getElementById('fileUpload').click();
}
function onCurrentPageBoxKeypress(event) {
if ((event.which < 48 || event.which > 57) && event.which !== 8 &&
event.which !== 13) {
event.preventDefault();
return false;
}
else {
var currentPageNumber = parseInt(currentPageBox.value);
if (event.which === 13) {
if (currentPageNumber > 0 && currentPageNumber <= viewer.pageCount) {
viewer.navigation.goToPage(currentPageNumber);
}
else {
currentPageBox.value = viewer.currentPageNumber.toString();
}
}
return true;
}
}

```

```

}
function onCurrentPageBoxClicked() {
currentPageBox.select();
currentPageBox.focus();
}
function updatePageNavigation() {
if (viewer.currentPageNumber === 1) {
toolbarObj.enableItems(document.getElementById('previousPage'), false);
toolbarObj.enableItems(document.getElementById('nextPage'), true);
}
else if (viewer.currentPageNumber === viewer.pageCount) {
toolbarObj.enableItems(document.getElementById('previousPage'), true);
toolbarObj.enableItems(document.getElementById('nextPage'), false);
}
else {
toolbarObj.enableItems(document.getElementById('previousPage'), true);
toolbarObj.enableItems(document.getElementById('nextPage'), true);
}
}
function updateZoomButtons() {
if (viewer.zoomPercentage <= 50) {
toolbarObj.enableItems(document.getElementById('zoomIn'), true);
toolbarObj.enableItems(document.getElementById('zoomOut'), false);
toolbarObj.enableItems(document.getElementById('fitPage'), true);
}
else if (viewer.zoomPercentage >= 400) {
toolbarObj.enableItems(document.getElementById('zoomIn'), false);
toolbarObj.enableItems(document.getElementById('zoomOut'), true);
toolbarObj.enableItems(document.getElementById('fitPage'), true);
}
else {
toolbarObj.enableItems(document.getElementById('zoomIn'), true);
toolbarObj.enableItems(document.getElementById('zoomOut'), true);
toolbarObj.enableItems(document.getElementById('fitPage'), true);
}
}
function nodeClick(args) {
var bookmarksDetails = viewer.bookmark.getBookmarks();
var bookmarksDestination = bookmarksDetails.bookmarksDestination;
var bookid = Number(args.nodeData.id);
var pageIndex = bookmarksDestination.bookMarkDestination[bookid].PageIndex;
var Y = bookmarksDestination.bookMarkDestination[bookid].Y;
viewer.bookmark.goToBookmark(pageIndex, Y);
return false;
}
function searchInputKeypressed(event) {
enablePrevButton(true);
enableNextButton(true);
if (event.which === 13) {
initiateTextSearch();
updateSearchInputIcon(false);
}
}
function searchClickHandler() {
if (searchButton.classList.contains('e-pv-search-icon')) {
viewer.textSearch.cancelTextSearch();
initiateTextSearch();
}
}

```

```
}
else if (searchButton.classList.contains('e-pv-search-close')) {
searchInput.value = '';
searchInput.focus();
viewer.textSearch.cancelTextSearch();
}
}
function initiateTextSearch() {
var searchString = searchInput.value;
viewer.textSearch.searchText(searchString, matchCase);
}
function previousSearchClicked() {
var searchString = searchInput.value;
if (searchString) {
viewer.textSearch.searchPrevious();
}
}
function nextSearchClicked() {
var searchString = searchInput.value;
if (searchString) {
viewer.textSearch.searchNext();
}
}
function checkBoxChanged(args) {
if (args.checked) {
matchCase = true;
}
else {
matchCase = false;
}
initiateTextSearch();
}
function enablePrevButton(isEnable) {
var previousSearchButton = document.getElementById('previousSearch');
if (isEnable) {
previousSearchButton.removeAttribute('disabled');
}
else {
previousSearchButton.disabled = true;
}
}
function enableNextButton(isEnable) {
var nextSearchButton = document.getElementById('nextSearch');
if (isEnable) {
nextSearchButton.removeAttribute('disabled');
}
else {
nextSearchButton.disabled = true;
}
}
function updateSearchInputIcon(isEnable) {
if (isEnable) {
searchButton.classList.remove('e-pv-search-close');
searchButton.classList.add('e-pv-search-icon');
}
else {
searchButton.classList.remove('e-pv-search-icon');
```

```

searchButton.classList.add('e-pv-search-close');
}
}
this.default = function () {
toolbarObj = new ej.navigations.Toolbar({
items: [
{ prefixIcon: 'e-pv-open-document', tooltipText: 'Open', id: 'openButton',
click: openClicked },
{ prefixIcon: 'e-pv-bookmark-icon', tooltipText: 'Bookmark', id:
'bookmarkButton', click: bookmarkClicked },
{ prefixIcon: 'e-pv-previous-page-navigation-icon', id: 'previousPage',
tooltipText: 'Previous Page', align: 'Center', click:
previousClicked.bind(this) },
{ prefixIcon: 'e-pv-next-page-navigation-icon', id: 'nextPage', tooltipText:
'Next Page', align: 'Center', click: nextClicked.bind(this) },
{ template: inputTemplate, tooltipText: 'Page Number', align: 'Center' },
{ template: totalPageNum, tooltipText: 'Page Number', align: 'Center' },
{ prefixIcon: 'e-pv-search-icon', tooltipText: 'Text Search', align:
'Right', click: searchClicked.bind(this) },
{ prefixIcon: 'e-pv-print-document-icon', tooltipText: 'Print', align:
'Right', click: printClicked.bind(this) },
{ prefixIcon: 'e-pv-download-document-icon', tooltipText: 'Download', align:
'Right', click: downloadClicked.bind(this) }
]
});
toolbarObj.appendTo('#topToolbar');
var magnificationToolbar = new ej.navigations.Toolbar({
items: [
{ prefixIcon: 'e-pv-fit-page-icon', id: 'fitPage', tooltipText: 'Fit to
page', click: pageFitClicked.bind(this) },
{ prefixIcon: 'e-pv-zoom-in-icon', id: 'zoomIn', tooltipText: 'Zoom in',
click: zoomInClicked.bind(this) },
{ prefixIcon: 'e-pv-zoom-out-icon', id: 'zoomOut', tooltipText: 'Zoom out',
click: zoomOutClicked.bind(this) },
]
});
magnificationToolbar.appendTo('#magnificationToolbar');
viewer = new ej.pdfviewer.PdfViewer({
enableToolbar: true,
enableThumbnail: true,
documentPath: 'https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
viewer.appendTo('#pdfViewer');
document.getElementById('fileUpload').addEventListener('change', readFile,
false);
currentPageBox = document.getElementById('currentPage');
currentPageBox.value = '1';
searchInput = document.getElementById('searchInput');
openDocument = document.getElementById('fileupload');
bookmarkPopup = new ej.popups.Dialog({
showCloseIcon: true, header: "Bookmarks", closeOnEscape: false, isModal:
false, target: document.getElementById('pdfViewer'),

```

```

content: '<div class="e-pv-bookmarks" id="bookmarkview"></div>',
buttons: [{
  buttonModel: {},
}], position: { X: 'left', Y: 'top' }, cssClass: 'e-bookmark-popup',
beforeClose: function () {
  isBookmarkOpen = false;
}
});
bookmarkPopup.appendTo('#popup');
textSearchPopup = new ej.popups.Dialog({
  showCloseIcon: false, closeOnEscape: false, isModal: false, target:
  document.getElementById('pdfViewer'),
  buttons: [{
    buttonModel: {},
  }], position: { X: 'right', Y: 'top' }, cssClass: 'e-text-search-popup',
});
textSearchPopup.appendTo('#textSearchBox');
var previousSearch = new ej.buttons.Button({ iconCss: 'e-pv-previous-search'
});
previousSearch.appendTo('#previousSearch');
var nextSearch = new ej.buttons.Button({ iconCss: 'e-pv-next-search-btn' });
nextSearch.appendTo('#nextSearch');
var matchCaseCheck = new ej.buttons.CheckBox({ label: 'Match case', change:
  checkBoxChanged });
matchCaseCheck.appendTo('#matchCase');
viewer.pageChange = function (args) {
  currentPageBox.value = viewer.currentPageNumber.toString();
  updatePageNavigation();
};
viewer.documentLoad = function (args) {
  document.getElementById('totalPage').textContent = 'of ' + viewer.pageCount;
  updatePageNavigation();
};
searchButton = document.getElementById('searchBtn');
searchInput.addEventListener('focus', function () {
  searchInput.parentElement.classList.add('e-input-focus'); });
searchInput.addEventListener('blur', function () {
  searchInput.parentElement.classList.remove('e-input-focus'); });
searchInput.addEventListener('keypress', searchInputKeypressed);
document.getElementById('previousSearch').addEventListener('click',
previousSearchClicked);
document.getElementById('nextSearch').addEventListener('click',
nextSearchClicked);
searchButton.addEventListener('click', searchClickHandler);
currentPageBox.addEventListener('keypress', onCurrentPageBoxKeyPress);
currentPageBox.addEventListener('click', onCurrentPageBoxClicked);
bookmarkPopup.hide();
textSearchPopup.hide();
enableNextButton(false);
enablePrevButton(false);
};
```

```

### **SERVER-BACKED**

```

```javascript

```

```

var inputTemplate = '<div class=""><input type="text" class="e-input-group e-pv-current-page-number" id="currentPage" /></div>';
var totalPageNum = '<div class=""><span class="e-pv-total-page-number" id="totalPage">of 0</span></div>';
var isBookmarkOpen = false;
var isBookmarkClick = false;
var isTextSearchBoxOpen = false;
var bookmarkPopup;
var textSearchPopup;
var toolbarObj;
var viewer;
var currentPageBox;
var searchInput;
var searchButton;
var openDocument;
var matchCase = false;
var fileInputElement;
var filename;
function previousClicked(args) {
hidePopups();
viewer.navigation.goToPreviousPage();
}
function hidePopups() {
isBookmarkOpen = false;
isTextSearchBoxOpen = false;
bookmarkPopup.hide();
textSearchPopup.hide();
}
function bookmarkClicked() {
textSearchPopup.hide();
if (!isBookmarkOpen) {
var bookmarkDetails = viewer.bookmark.getBookmarks();
if (bookmarkDetails.bookmarks) {
var bookmarks = bookmarkDetails.bookmarks.bookMark;
var treeObj = new ej.navigations.TreeView({
fields: {
dataSource: bookmarks,
id: 'Id',
parentID: 'Pid',
text: 'Title',
hasChildren: 'HasChild',
}, nodeSelected: nodeClick
});
treeObj.appendTo('#bookmarkview');
bookmarkPopup.show();
isBookmarkOpen = true;
isBookmarkClick = true;
}
else {
toolbarObj.enableItems(document.getElementById('bookmarkButton'), false);
isBookmarkOpen = false;
}
}
else {
if (!isBookmarkClick) {
bookmarkPopup.show();
isBookmarkClick = true;
}
}
}

```



```
    } else {
        bookmarkPopup.hide();
        isBookmarkClick = false;
    }
}

function nextClicked(args) {
    hidePopups();
    viewer.navigation.goToNextPage();
}

function searchClicked(args) {
    bookmarkPopup.hide();
    if (!isTextSearchBoxOpen) {
        textSearchPopup.show();
    }
    else {
        viewer.textSearch.cancelTextSearch();
        textSearchPopup.hide();
    }
    isTextSearchBoxOpen = !isTextSearchBoxOpen;
}

function printClicked(args) {
    hidePopups();
    viewer.print.print();
}

function downloadClicked(args) {
    hidePopups();
    viewer.download();
}

function pageFitClicked(args) {
    hidePopups();
    viewer.magnification.fitToPage();
    updateZoomButtons();
    toolbarObj.enableItems(document.getElementById('fitPage'), false);
}

function zoomInClicked(args) {
    hidePopups();
    viewer.magnification.zoomIn();
    updateZoomButtons();
}

function zoomOutClicked(args) {
    hidePopups();
    viewer.magnification.zoomOut();
    updateZoomButtons();
}

function readFile(args) {
    // tslint:disable-next-line
    var uploadedFiles = args.target.files;
    if (args.target.files[0] !== null) {
        var uploadedFile = uploadedFiles[0];
        filename = uploadedFiles[0].name;
        if (uploadedFile) {
            var reader = new FileReader();
            reader.readAsDataURL(uploadedFile);
            // tslint:disable-next-line
            reader.onload = function (e) {
                var uploadedFileUrl = e.currentTarget.result;
```

```
viewer.load(uploadedFileUrl, null);
currentPageBox.value = '1';
document.getElementById("bookmarkview").innerHTML = "";
isBookmarkOpen = false;
viewer.fileName = filename;
};
}
}
}
function openClicked() {
document.getElementById('fileUpload').click();
}
function onCurrentPageBoxKeypress(event) {
if ((event.which < 48 || event.which > 57) && event.which !== 8 &&
event.which !== 13) {
event.preventDefault();
return false;
}
else {
var currentPageNumber = parseInt(currentPageBox.value);
if (event.which === 13) {
if (currentPageNumber > 0 && currentPageNumber <= viewer.pageCount) {
viewer.navigation.goToPage(currentPageNumber);
}
else {
currentPageBox.value = viewer.currentPageNumber.toString();
}
}
return true;
}
}
function onCurrentPageBoxClicked() {
currentPageBox.select();
currentPageBox.focus();
}
function updatePageNavigation() {
if (viewer.currentPageNumber === 1) {
toolbarObj.enableItems(document.getElementById('previousPage'), false);
toolbarObj.enableItems(document.getElementById('nextPage'), true);
}
else if (viewer.currentPageNumber === viewer.pageCount) {
toolbarObj.enableItems(document.getElementById('previousPage'), true);
toolbarObj.enableItems(document.getElementById('nextPage'), false);
}
else {
toolbarObj.enableItems(document.getElementById('previousPage'), true);
toolbarObj.enableItems(document.getElementById('nextPage'), true);
}
}
function updateZoomButtons() {
if (viewer.zoomPercentage <= 50) {
toolbarObj.enableItems(document.getElementById('zoomIn'), true);
toolbarObj.enableItems(document.getElementById('zoomOut'), false);
toolbarObj.enableItems(document.getElementById('fitPage'), true);
}
else if (viewer.zoomPercentage >= 400) {
toolbarObj.enableItems(document.getElementById('zoomIn'), false);
```

```

toolbarObj.enableItems(document.getElementById('zoomOut'), true);
toolbarObj.enableItems(document.getElementById('fitPage'), true);
}
else {
toolbarObj.enableItems(document.getElementById('zoomIn'), true);
toolbarObj.enableItems(document.getElementById('zoomOut'), true);
toolbarObj.enableItems(document.getElementById('fitPage'), true);
}
}
function nodeClick(args) {
var bookmarksDetails = viewer.bookmark.getBookmarks();
var bookmarksDestination = bookmarksDetails.bookmarksDestination;
var bookid = Number(args.nodeData.id);
var pageIndex = bookmarksDestination.bookMarkDestination[bookid].PageIndex;
var Y = bookmarksDestination.bookMarkDestination[bookid].Y;
viewer.bookmark.goToBookmark(pageIndex, Y);
return false;
}
function searchInputKeypressed(event) {
enablePrevButton(true);
enableNextButton(true);
if (event.which === 13) {
initiateTextSearch();
updateSearchInputIcon(false);
}
}
function searchClickHandler() {
if (searchButton.classList.contains('e-pv-search-icon')) {
viewer.textSearch.cancelTextSearch();
initiateTextSearch();
}
else if (searchButton.classList.contains('e-pv-search-close')) {
searchInput.value = '';
searchInput.focus();
viewer.textSearch.cancelTextSearch();
}
}
function initiateTextSearch() {
var searchString = searchInput.value;
viewer.textSearch.searchText(searchString, matchCase);
}
function previousSearchClicked() {
var searchString = searchInput.value;
if (searchString) {
viewer.textSearch.searchPrevious();
}
}
function nextSearchClicked() {
var searchString = searchInput.value;
if (searchString) {
viewer.textSearch.searchNext();
}
}
function checkBoxChanged(args) {
if (args.checked) {
matchCase = true;
}
}

```

```

else {
  matchCase = false;
}
initiateTextSearch();
}
function enablePrevButton(isEnable) {
  var previousSearchButton = document.getElementById('previousSearch');
  if (isEnable) {
    previousSearchButton.removeAttribute('disabled');
  }
  else {
    previousSearchButton.disabled = true;
  }
}
function enableNextButton(isEnable) {
  var nextSearchButton = document.getElementById('nextSearch');
  if (isEnable) {
    nextSearchButton.removeAttribute('disabled');
  }
  else {
    nextSearchButton.disabled = true;
  }
}
function updateSearchInputIcon(isEnable) {
  if (isEnable) {
    searchButton.classList.remove('e-pv-search-close');
    searchButton.classList.add('e-pv-search-icon');
  }
  else {
    searchButton.classList.remove('e-pv-search-icon');
    searchButton.classList.add('e-pv-search-close');
  }
}
this.default = function () {
  toolbarObj = new ej.navigations.Toolbar({
    items: [
      { prefixIcon: 'e-pv-open-document', tooltipText: 'Open', id: 'openButton',
        click: openClicked },
      { prefixIcon: 'e-pv-bookmark-icon', tooltipText: 'Bookmark', id:
        'bookmarkButton', click: bookmarkClicked },
      { prefixIcon: 'e-pv-previous-page-navigation-icon', id: 'previousPage',
        tooltipText: 'Previous Page', align: 'Center', click:
        previousClicked.bind(this) },
      { prefixIcon: 'e-pv-next-page-navigation-icon', id: 'nextPage', tooltipText:
        'Next Page', align: 'Center', click: nextClicked.bind(this) },
      { template: inputTemplate, tooltipText: 'Page Number', align: 'Center' },
      { template: totalPageNum, tooltipText: 'Page Number', align: 'Center' },
      { prefixIcon: 'e-pv-search-icon', tooltipText: 'Text Search', align:
        'Right', click: searchClicked.bind(this) },
      { prefixIcon: 'e-pv-print-document-icon', tooltipText: 'Print', align:
        'Right', click: printClicked.bind(this) },
      { prefixIcon: 'e-pv-download-document-icon', tooltipText: 'Download', align:
        'Right', click: downloadClicked.bind(this) }
    ]
  });
  toolbarObj.appendTo('#topToolbar');
  var magnificationToolbar = new ej.navigations.Toolbar({

```

```

items: [
  { prefixIcon: 'e-pv-fit-page-icon', id: 'fitPage', tooltipText: 'Fit to
page', click: pageFitClicked.bind(this) },
  { prefixIcon: 'e-pv-zoom-in-icon', id: 'zoomIn', tooltipText: 'Zoom in',
click: zoomInClicked.bind(this) },
  { prefixIcon: 'e-pv-zoom-out-icon', id: 'zoomOut', tooltipText: 'Zoom out',
click: zoomOutClicked.bind(this) },
]
});
magnificationToolbar.appendTo('#magnificationToolbar');
viewer = new ej.pdfviewer.PdfViewer({
  enableToolbar: true,
  enableThumbnail: true,
  documentPath: 'https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf',
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer/'
});
ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar,
ej.pdfviewer.Magnification,
ej.pdfviewer.LinkAnnotation, ej.pdfviewer.ThumbnailView,
ej.pdfviewer.BookmarkView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Navigation, ej.pdfviewer.Print);
viewer.appendTo('#pdfViewer');
document.getElementById('fileUpload').addEventListener('change', readFile,
false);
currentPageBox = document.getElementById('currentPage');
currentPageBox.value = '1';
searchInput = document.getElementById('searchInput');
openDocument = document.getElementById('fileupload');
bookmarkPopup = new ej.popups.Dialog({
  showCloseIcon: true, header: "Bookmarks", closeOnEscape: false, isModal:
false, target: document.getElementById('pdfViewer'),
  content: '<div class="e-pv-bookmarks" id="bookmarkview"></div>',
  buttons: [{
    buttonModel: {},
  }], position: { X: 'left', Y: 'top' }, cssClass: 'e-bookmark-popup',
  beforeClose: function () {
    isBookmarkOpen = false;
  }
});
bookmarkPopup.appendTo('#popup');
textSearchPopup = new ej.popups.Dialog({
  showCloseIcon: false, closeOnEscape: false, isModal: false, target:
document.getElementById('pdfViewer'),
  buttons: [{
    buttonModel: {},
  }], position: { X: 'right', Y: 'top' }, cssClass: 'e-text-search-popup',
});
textSearchPopup.appendTo('#textSearchBox');
var previousSearch = new ej.buttons.Button({ iconCss: 'e-pv-previous-search'
});
previousSearch.appendTo('#previousSearch');
var nextSearch = new ej.buttons.Button({ iconCss: 'e-pv-next-search-btn' });
nextSearch.appendTo('#nextSearch');
var matchCaseCheck = new ej.buttons.CheckBox({ label: 'Match case', change:
checkBoxChanged });
matchCaseCheck.appendTo('#matchCase');
viewer.pageChange = function (args) {

```

```

currentPageBox.value = viewer.currentPageNumber.toString();
updatePageNavigation();
};
viewer.documentLoad = function (args) {
document.getElementById('totalPage').textContent = 'of ' + viewer.pageCount;
updatePageNavigation();
};
searchButton = document.getElementById('searchBtn');
searchInput.addEventListener('focus', function () {
searchInput.parentElement.classList.add('e-input-focus'); });
searchInput.addEventListener('blur', function () {
searchInput.parentElement.classList.remove('e-input-focus'); });
searchInput.addEventListener('keypress', searchInputKeypressed);
document.getElementById('previousSearch').addEventListener('click',
previousSearchClicked);
document.getElementById('nextSearch').addEventListener('click',
nextSearchClicked);
searchButton.addEventListener('click', searchClickHandler);
currentPageBox.addEventListener('keypress', onCurrentPageBoxKeypress);
currentPageBox.addEventListener('click', onCurrentPageBoxClicked);
bookmarkPopup.hide();
textSearchPopup.hide();
enableNextButton(false);
enablePrevButton(false);
};
`ts

```

Sample :

<https://ej2.syncfusion.com/javascript/demos/#/material/pdfviewer/custom-toolbar.html>

Highlight underline strikethrough text in EJ2 JavaScript Pdfviewer control

The PDF Viewer library allows you to highlight, underline and strikethrough text in the loaded PDF document programmatically using the **setAnnotationMode()** method.

Step 1: Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

Step 2: Add the following code snippet to highlight, underline, and strikethrough text in index.ts file with button click events.

```

`ts
document.getElementById('setHighlight').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Highlight');
});
document.getElementById('setUnderline').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Underline');
});
document.getElementById('setStrikethrough').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Strikethrough');
});

```

```
document.getElementById('setNone').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('None');
});
`ts
```

Find the Sample [how to highlight, underline and strikethrough text programmatically](#)

Enable resize in EJ2 JavaScript Pdfviewer control

To enable the resizer for the text markup annotation in Syncfusion PDF viewer, you can use the **enableTextMarkupResizer** property. Default value of the property is false.

Here is an example of how you can enable the resizer for the text markup annotation:

```
`ts
//Enable TextMarkup Resizer.
viewer.enableTextMarkupResizer = true;
`ts
```

Find the sample [how to enable the resizer for the Text Markup annotation](#)

Customize text search color in EJ2 JavaScript Pdfviewer control

To change the text search color in the Syncfusion PDF viewer, you can use the **searchColor** property of the searchModule object. This property accepts a string value that represents the color in hexadecimal format.

```
`ts
viewer.textSearchColorSettings.searchColor = "#FF0000";
`ts
```

This will set the text search color to red. You can use any valid hexadecimal color code to set the text search color to the desired color.

You can also use the **searchHighlightColor** property of the searchModule object to change the highlight color of the search results. This property also accepts a string value in hexadecimal format.

```
`ts
viewer.textSearchColorSettings.searchHighlightColor = "#0000FF";
`ts
```

This will set the highlight color of the search results to blue.

- [searchColor](#)
- [searchHighlightColor](#)

Here is an example of how you can use the searchHighlightColor property and searchColor property:

```
`ts
<button id="search">SearchText</button>
<button id="searchNext">SearchNext</button>
`ts
```

```

<button id="searchPervious">searchPervious</button>
<button id="searchCancel">CancelSearch</button>
,
`ts
viewer.enableTextSearch = true;
// customize the textSeach color
viewer.textSearchColorSettings.searchColor = "#FF0000";
// customize the highlight color of the search results
viewer.textSearchColorSettings.searchHighlightColor = "#0000FF";
document.getElementById("search").addEventListener("click", () => {
viewer.textSearchModule.searchText("pdf", false);
});
document.getElementById("searchNext").addEventListener("click", () => {
viewer.textSearchModule.searchNext();
});
document.getElementById("searchPervious").addEventListener("click", () => {
viewer.textSearchModule.searchPrevious();
});
document.getElementById("searchCancel").addEventListener("click", () => {
viewer.textSearchModule.cancelTextSearch();
});
,

```

Find the sample [how to customize the text search color](#)

Disable context menu in EJ2 JavaScript Pdfviewer control

To disable the context menu in the Syncfusion PDF viewer control, you can use the **ContextMenuOption** property as **'None'** to hide all context menu options. Default value of the **ContextMenuOption** is **'RightClick'**.

Here is an example of how you can use the **ContextMenuOption** to disable context menu in the PDF Viewer:

```

,
<button id='disable'>Disable ContextMenuOption</button>
,
`ts
// Disable ContextMenuOption

```



```
document.getElementById('disable').addEventListener('click', ()=> {
viewer.contextMenuOption = 'None';
});
`ts
```

This will hide the context menu and prevent the user from right-clicking on the PDF viewer.

Find the sample [how to disable Context Menu](#)

Extract texts in EJ2 JavaScript Pdfviewer control

To extract text in Syncfusion PDF viewer, you can use the **isExtractText** property and **extractTextCompleted** event. This allows you to extract the text from a page along with the bounds.

Here is an example of how you can use the **isExtractText** property and **extractTextCompleted** event:

```
`ts
viewer.isExtractText = true;
viewer.extractTextCompleted = args => {
//Extract the Complete text of load document
console.log(args);
console.log(args.documentTextCollection[1]);
//Extract the Text data.
console.log(args.documentTextCollection[1][1].textData);
//Extract Text in the Page.
console.log(args.documentTextCollection[1][1].pageText);
//Extracts the first text of the PDF document along with its bounds
console.log(args.documentTextCollection[1][1].textData[0].Bounds);
};
`ts
```

Find the sample [how to extract Text](#)

Import annotations in EJ2 JavaScript Pdfviewer control

To import annotations into a Syncfusion PDF Viewer when loading a PDF document, you can use the **importAnnotations()** method of the PDF Viewer. This method allows you to import annotations from a file or a string in the PDF Viewer.

Here is an example of how you can use the **importAnnotations()** method to import annotations when loading a PDF document:

Step 1: Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

Step 2: Add the following code snippet to import annotations on load PDF document.

```
`ts
document.getElementById('importAnnot').addEventListener('click', ()=> {
```

```
viewer.importAnnotation({
  pdfAnnotation: {
    '0': {
      shapeAnnotation: [
        {
          ShapeAnnotationType: 'Square',
          Author: 'Guest',
          AnnotationSelectorSettings: {
            selectionBorderColor: '',
            resizerBorderColor: 'black',
            resizerFillColor: '#FF4081',
            resizerSize: 8,
            selectionBorderThickness: 1,
            resizerShape: 'Square',
            selectorLineDashArray: [],
            resizerLocation: 3,
            resizerCursorType: null
          },
          ModifiedDate: '4/22/2021, 10:33:04 AM',
          Subject: 'Rectangle',
          Note: '',
          IsCommentLock: false,
          StrokeColor: 'rgba(255,0,0,1)',
          FillColor: 'rgba(255,255,255,0)',
          Opacity: 1,
          Bounds: {
            X: 124,
            Y: 76,
            Width: 202,
            Height: 154,
            Location: { X: 124, Y: 76 },
            Size: { IsEmpty: false, Width: 202, Height: 154 },
            Left: 124,
```

Top: 76,
Right: 326,
Bottom: 230
,
Thickness: 2,
BorderStyle: 'Solid',
BorderDashArray: 0,
RotateAngle: 'RotateAngle0',
IsCloudShape: false,
CloudIntensity: 0,
RectangleDifference: null,
VertexPoints: null,
LineHeadStart: null,
LineHeadEnd: null,
IsLocked: false,
AnnotName: 'e9a14dbe-5d09-4226-329e-c6edab201284',
Comments: null,
State: "",
StateModel: "",
AnnotType: 'shape',
EnableShapeLabel: false,
LabelContent: null,
LabelFillColor: null,
LabelBorderColor: null,
FontColor: null,
FontSize: 0,
CustomData: null,
LabelBounds: {
X: 0,
Y: 0,
Width: 0,
Height: 0,
Location: { X: 0, Y: 0 },

```

Size: { IsEmpty: true, Width: 0, Height: 0 },
Left: 0,
Top: 0,
Right: 0,
Bottom: 0
},
LabelSettings: null,
AnnotationSettings: {
  minWidth: 0,
  maxWidth: 0,
  minHeight: 0,
  maxHeight: 0,
  isLock: false,
  isPrint: true
},
AllowedInteractions: ['None'],
IsPrint: true,
ExistingCustomData: null
},
]
}
}
});
});
,

```

In this example, the PDF Viewer is first used to load the PDF document from a file. The **importAnnotations()** method is then used to import annotations from a string. The imported annotations will be added to the PDF Viewer and displayed along with the document.

Alternatively, we can import annotations from a file in JSON or XFDF format.

```

`ts
document.getElementById('import').addEventListener('click', function() {
viewer.importAnnotation('F# Succinctly.xfdf');
});
,

```

Find the sample [how to import annotations on loading a PDF Document](#)

Identify added annotation mode in EJ2 JavaScript Pdfviewer control

The PDF Viewer library allows you to identify whether the added annotations in PDF document is UI drawn, imported or existing annotation. Annotation mode can be identified using the **annotationAddMode** property of **annotationSelect** event.

Step 1: Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

Step 2: The following code snippet explains how to identify added annotation mode.

```
`ts
viewer.annotationSelect =(args) =>{
console.log(args.annotationAddMode);
}
`
```

Find the Sample [how to identify added annotation mode](#)

Overlapped annotation in EJ2 JavaScript Pdfviewer control

To get the overlapped annotations on a mouse click in a Syncfusion PDF Viewer, you can use the **annotationCollection** property of **annotationSelect** event. This event is triggered when the user clicks on an annotation in the PDF document.

Here is an example of how you can use the **annotationSelect** event to get the overlapped annotations on a mouse click in a Syncfusion PDF Viewer:

```
`ts
// Get overlapped annotation collections.
viewer.annotationSelect =(args) =>{
console.log(args.annotationCollection);
}
`
```

Find the sample [how to get the overlapped annotations on mouse click](#)

Display custom tool tip for annotation in EJ2 JavaScript Pdfviewer control

To display a custom tooltip for annotations using the mouseover event in the Syncfusion PDF Viewer, you can use the **annotationMouseover** and **annotationMouseLeave** events .

Here is an example that demonstrates how to display a custom tooltip for annotations using the mouseover event in the Syncfusion PDF Viewer:

```
`ts
viewer.annotationSettings.author = "syncfusion";
let tooltip: Tooltip = new Tooltip({
mouseTrail: true,
opensOn: "Custom"
```

```

});
viewer.annotationMouseOver = args => {
  tooltip.appendTo("#pdfViewerpageDiv" + (viewer.currentPageNumber - 1));
  tooltip.content = args.annotation.author;
  tooltip.open();
  let ele: any = document.getElementsByClassName("e-tooltip-wrap")[0];
  ele.style.top = args.Y + 100 + "px";
  ele.style.left = args.X + "px";
};
viewer.annotationMouseLeave = args => {
  tooltip.close();
};
`

```

Find the sample [how to display custom tooltip for annotations using MouseOver event](#)

Select multi page annotations in EJ2 JavaScript Pdfviewer control

To select a multi-page TextMarkup annotation as a single annotation in a Syncfusion PDF viewer, you can use by enabling the **enableMultiPageAnnotation** property. By default it is **false**.

Here is an example of how you can use the **enableMultiPageAnnotation** property to select the multi page TextMarkup annotation as a single annotation, export and import multi page annotation:

```

`ts
// Enable multi-page TextMarkup Annotation.
viewer.enableMultiPageAnnotation = true;
// Export Annotation
document.getElementById('export').addEventListener('click', () => {
  viewer.exportAnnotation();
});
// Import Annotation.
document.getElementById('import').addEventListener('click', () => {
  viewer.importAnnotation("Add Export annotation file content");
});
`

```

Find the sample [how to select multi-page TextMarkup annotation as single annotation](#)

Disable tile rendering in EJ2 JavaScript Pdfviewer control

To disable the tile rendering feature in the Syncfusion PDF viewer control, you can use the **enableTileRendering** property. This property allows you to enable or disable the tile rendering feature, which is used to improve the performance of the PDF viewer when displaying large documents.

To disable the tile rendering feature, you can set the **enableTileRendering** property to **false**.

By default, the tile rendering feature is enabled in the PDF viewer. Disabling it may improve the performance of the PDF viewer when displaying small documents, but it may also reduce the performance when displaying large documents.

Here is an example of how you can use the **enableTileRendering** property:

```
,
<button id="disable">disable tile rendering</button>
,
`ts
// Disable tile rendering.
document.getElementById('disable').addEventListener('click', () => {
viewer.tileRenderingSettings.enableTileRendering = false;
});
,
```

Find the sample [how to disable the tile rendering](#)

Add header value in EJ2 JavaScript Pdfviewer control

To add header values to an AJAX request made by a Syncfusion PDF Viewer, you can use the **ajaxHeaders** property available in the **ajaxRequestSettings** module of the PDF Viewer. This property allows you to specify custom headers for the AJAX request.

Here is an example of how you can use the **ajaxRequestSettings** property to add a custom header to an AJAX request made by the PDF Viewer:

```
`ts
viewer.ajaxRequestSettings = {
  ajaxHeaders: [
    {
      headerName: "Authorization",
      headerValue: "Bearer 64565dfgfdsjweiuvbiuyhiueygf"
    }
  ],
  withCredentials: false
};
,
```

Find the sample [how to add custom headers in AjaxRequestSettings](#)

Print document in EJ2 JavaScript Pdfviewer control

The PDF Viewer library allows you to print the PDF document programmatically using the **print()** method in the **PrintModule**.

The following steps are used to print the PDF document programmatically.

Step 1: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Step 2: Add the following code snippet to perform the print operation.

```
,  
  
<button id="print">Print</button>  
  
,  
  
`ts  
document.getElementById('print').addEventListener('click', ()=> {  
//Print the loaded document.  
viewer.printModule.print();  
});  
,
```

Find the Sample, [how to print the PDF document programmatically](#)

Unload document in EJ2 JavaScript Pdfviewer control

The PDF Viewer library allows you to unload the PDF document being display in the PDF Viewer control programmatically using the [unload\(\)](#) method.

The following steps are used to unload the PDF document programmatically.

Step 1: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Step 2: Add the following code snippet to perform the unload operation.

```
,  
  
<button id="unload">Unload Document</button>  
  
,  
  
`ts  
document.getElementById('unload').addEventListener('click', () => {  
// Unload the document.  
viewer.unload();  
});  
,
```

Find the Sample, [how to unload the PDF document programmatically](#)

Load document in EJ2 JavaScript Pdfviewer control

The PDF Viewer library allows to switch or load the PDF documents dynamically after the initial load operation. To achieve this, load the PDF document as a base64 string or file name in PDF Viewer control using the [Load\(\)](#) method dynamically.

The following steps are used to load the PDF document dynamically.

Step 1: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Step 2: Use the following code snippet to load PDF document using base64 string.

```
<button id='load1'>LoadDocumentFromBase64</button>

`ts
// Load PDF document from Base64 string
document.getElementById('load1').addEventListener('click', () => {
viewer.load(
'data:application/pdf;base64,'+ AddBase64String, null);
})
```

Step 3: Use the following code snippet to load PDF document using document name.

```
<button id='load2'>LoadDocument</button>

`ts
// Load PDF document using file name
document.getElementById('load2').addEventListener('click', () => {
viewer.load('https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf', null);
});
```

Find the sample, [how to load PDF documents dynamically](#)

Import export annotation object in EJ2 JavaScript Pdfviewer control

The PDF Viewer library allows you to import annotations from objects or streams instead of loading it as a file. To import such annotation objects, the PDF Viewer control must export the PDF annotations as objects using the [ExportAnnotationsAsObject\(\)](#) method. Only the annotations objects that are exported from the PDF Viewer can be imported.

The following steps are used to import and export annotation as object.

Step 1: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Step 2: Use the following code snippet to perform import and export annotation.

```
,
<button id="export">Export Annotation</button>
<button id="import">Import Annotation</button>
,
`ts
//Export annotation as object.
document.getElementById('export').addEventListener('click', () => {
viewer.exportAnnotationsAsObject().then(function(value) {
exportObject = value;
});
});
//Import annotation that are exported as object.
document.getElementById('import').addEventListener('click', () => {
viewer.importAnnotation(JSON.parse(exportObject));
});
,
```

Find the Sample, [how to import and export annotation as object](#)

Delete annotation in EJ2 JavaScript Pdfviewer control

The PDF Viewer library allows you to delete a specific annotation from a PDF document. Deleting a specific annotation can be done using the **deleteAnnotationById()** method. This method is used to delete a specific annotation using its id.

The following steps are used to delete a specific annotation from PDF Document.

Step 1: Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

Step 2: Use the following code snippet to delete a specific annotation using `deleteAnnotationById()` method.

```
,
<button id="deleteAnnotationbyId">Delete Annotation By Id</button>
,
`ts
// Delete Annotation by ID.
document.getElementById('deleteAnnotationbyId').addEventListener('click', () => {
viewer.annotationModule.deleteAnnotationById(
viewer.annotationCollection[0].annotationId
```

```
);
});
`
```

Find the sample [how to delete a specific annotation using deleteAnnotationById](#)

Open thumbnail in EJ2 JavaScript Pdfviewer control

The PDF Viewer library allows you to open the thumbnail pane programmatically using the [openThumbnailPane\(\)](#) method.

The following steps are used to open the thumbnail.

Step 1: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Step 2: Use the following code snippet to open thumbnail.

```
`
<button id="openThumbnail">Open Thumbnail Pane</button>
`
`ts
document.getElementById('openThumbnail').addEventListener('click', () => {
// Open Thumbnail pane
viewer.thumbnailViewModule.openThumbnailPane();
});
`
```

Find the sample, [how to open the thumbnail pane programmatically](#)

Annotation selectors in EJ2 JavaScript Pdfviewer control

To customize the annotation selector in Syncfusion PDF Viewer, you can use the **annotationSelectorSettings** property of the PdfViewer control.

Here is an example of how you can customize the selector of the shape annotation:

```
`
<button id="annotationSelector">annotationSelector</button>
`
`ts
document.getElementById('annotationSelector').addEventListener('click', () => {
viewer.rectangleSettings.annotationSelectorSettings.resizerShape = 'Circle';
viewer.annotationModule.editAnnotation(viewer.annotationCollection[0]);
});
`
```

Find the sample [how to customize the annotation selector](#)

Clear annotations in EJ2 JavaScript Pdfviewer control

To clear all the annotations in a PDF document using the Syncfusion PDF Viewer, you can use the **deleteAnnotations** method. This method is used to clear all the annotations present in the currently loaded document.

Here is an example of how you can clear all the annotations present in the currently loaded document:

```
,
<button id="deleteAnnotations">Delete Annotations</button>
,
`ts
//clear Annotations.
document.getElementById('deleteAnnotations').addEventListener('click',()=> {
viewer.deleteAnnotations();
})
,
```

We can also delete specific annotations with the **deleteAnnotationById()** method. This method is used to delete a specific annotation using its id.

Here is an example of how you can delete specific annotations with the **deleteAnnotationById** method:

```
,
<button id="deleteAnnotationbyId">Delete Annotation By Id</button>
,
`ts
//Delete Annotation by ID.
document.getElementById('deleteAnnotationbyId').addEventListener('click', () => {
viewer.annotationModule.deleteAnnotationById(
viewer.annotationCollection[0].annotationId
);
});
,
```

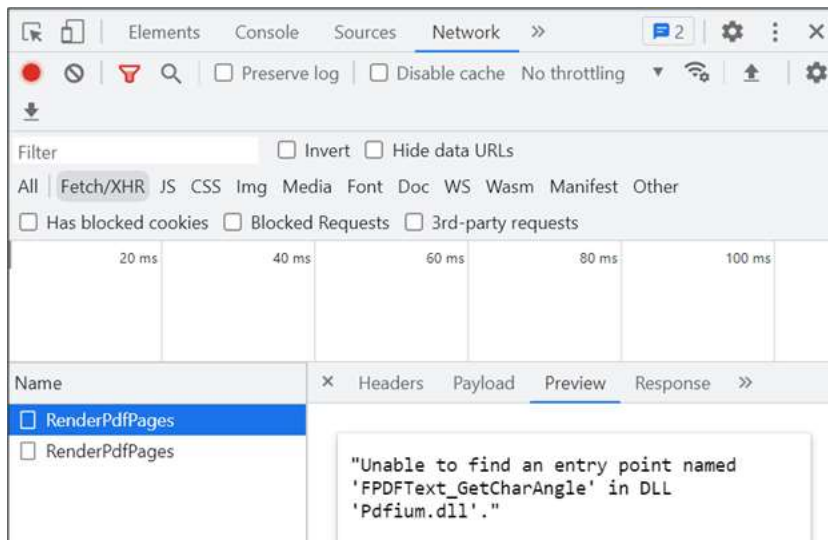
Find the sample [how to clear annotations using deleteAnnotations](#)

Resolve unable to find an entry point error in EJ2 JavaScript Pdfviewer control

From the release of version **21.1.0.35 (2023 Volume 1)** of Essential Studio, the Pdfium package has been upgraded to improve various functionalities like text search, text selection, rendering, and even performance. If you are updating your project to this version of the Syncfusion PDF Viewer, you may encounter the “**Web-Service is not listening**” error. The Network tab can help you identify the root cause of the issue, which is typically caused by an old version of pdfium assembly being referenced in

the local web service project. Below are the assemblies to be referred to in the respective operating systems.

- Windows – pdfium.dll
- Linux – libpdfium.so
- OSX – libpdfium.dylib



To solve this issue, you should follow the below steps

1. Clear the bin and object files of the web-service project.
2. Re-publish the web-service project.

Note: Note: If you are hosting your application in Azure, AWS, or in Linux environments, delete the older published files and republish the application.

Customize toolbar in PDF Viewer component

How to customize existing toolbar in PDF Viewer component

PDF Viewer allows you to customize(add, show, hide, enable, and disable) existing items in a toolbar.

- Add - New items can be defined by [CustomToolbarItemModel](#) and with existing items in [ToolbarSettings](#) property. Newly added item click action can be defined in [toolbarclick](#).
- Show, Hide - Existing items can be shown or hidden using the [ToolbarSettings](#) property. Pre-defined toolbar items are available with [ToolbarItem](#).
- Enable, Disable - Toolbar items can be enabled or disable using [enabletoolbaritem](#)

STANDALONE

```
import { PdfViewer, Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, ThumbnailView, BookmarkView, TextSelection, TextSearch,
FormFields, FormDesigner, Print, CustomToolbarItemModel } from
"@syncfusion/ej2-pdfviewer";
import { ComboBox } from "@syncfusion/ej2-dropdowns";
import { TextBox } from "@syncfusion/ej2-inputs";
```

```

PdfViewer.Inject(Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, ThumbnailView, BookmarkView, TextSelection, TextSearch,
FormFields, FormDesigner, Print);
let toolItem1: CustomToolbarItemModel = {
prefixIcon: 'e-icons e-paste',
id: 'print',
tooltipText: 'Custom toolbar item',
};
let toolItem2: CustomToolbarItemModel = {
id: 'download',
text: 'Save',
tooltipText: 'Custom toolbar item',
align: 'right'
};
let LanguageList: string[] = ['Typescript', 'Javascript', 'Angular', 'C#',
'C', 'Python'];
let toolItem3: CustomToolbarItemModel = {
type: 'Input',
tooltipText: 'Language List',
cssClass: 'percentage',
align: 'Left',
id: 'dropdown',
template: new ComboBox({ width: 100, value: 'TypeScript', dataSource:
LanguageList, popupWidth: 85, showClearButton: false, readonly: false })
};
let toolItem4: CustomToolbarItemModel = {
type: 'Input',
tooltipText: 'Text',
align: 'Right',
cssClass: 'find',
id: 'textbox',
template: new TextBox({ width: 125, placeholder: 'Type Here', created:
OnCreateSearch })
}
let pdfviewer: PdfViewer = new PdfViewer();
pdfviewer.documentPath = "https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf";
pdfviewer.resourceUrl = "https://cdn.syncfusion.com/ej2/23.1.43/dist/ej2-
pdfviewer-lib";
pdfviewer.toolbarSettings = { toolbarItems: [toolItem1, toolItem2,
'OpenOption', 'PageNavigationTool', 'MagnificationTool', toolItem3,
'PanTool', 'SelectionTool', 'SearchOption', 'PrintOption', 'DownloadOption',
'UndoRedoTool', 'AnnotationEditTool', 'FormDesignerEditTool', toolItem4,
'CommentTool', 'SubmitForm']}
pdfviewer.appendTo('#PdfViewer');
//To handle custom toolbar click event.
pdfviewer.toolbarClick = function (args) {
if (args.item && args.item.id === 'print') {
pdfviewer.print.print();
}
else if (args.item && args.item.id === 'download') {
pdfviewer.download();
}
};
function OnCreateSearch(this: any): any {
this.addIcon('prepend', 'e-icons e-search');
}

```

SERVER-BACKED

```

import { PdfViewer, Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, ThumbnailView, BookmarkView, TextSelection, TextSearch,
FormFields, FormDesigner, Print, CustomToolbarItemModel } from
'@syncfusion/ej2-pdfviewer';
import { ComboBox } from "@syncfusion/ej2-dropdowns";
import { TextBox } from "@syncfusion/ej2-inputs";
PdfViewer.Inject(Toolbar, Magnification, Navigation, Annotation,
LinkAnnotation, ThumbnailView, BookmarkView, TextSelection, TextSearch,
FormFields, FormDesigner, Print);
let toolItem1: CustomToolbarItemModel = {
prefixIcon: 'e-icons e-paste',
id: 'print',
tooltipText: 'Custom toolbar item',
};
let toolItem2: CustomToolbarItemModel = {
id: 'download',
text: 'Save',
tooltipText: 'Custom toolbar item',
align: 'right'
};
let LanguageList: string[] = ['Typescript', 'Javascript', 'Angular', 'C#',
'C', 'Python'];
let toolItem3: CustomToolbarItemModel = {
type: 'Input',
tooltipText: 'Language List',
cssClass: 'percentage',
align: 'Left',
id: 'dropdown',
template: new ComboBox({ width: 100, value: 'TypeScript', dataSource:
LanguageList, popupWidth: 85, showClearButton: false, readonly: false })
};
let toolItem4: CustomToolbarItemModel = {
type: 'Input',
tooltipText: 'Text',
align: 'Right',
cssClass: 'find',
id: 'textbox',
template: new TextBox({ width: 125, placeholder: 'Type Here', created:
OnCreateSearch })
}
let pdfviewer: PdfViewer = new PdfViewer();
pdfviewer.documentPath = "https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf";
pdfviewer.serviceUrl =
'https://services.syncfusion.com/js/production/api/pdfviewer';
pdfviewer.toolbarSettings = { toolbarItems: [toolItem1, toolItem2,
'OpenOption', 'PageNavigationTool', 'MagnificationTool', toolItem3,
'PanTool', 'SelectionTool', 'SearchOption', 'PrintOption', 'DownloadOption',
'UndoRedoTool', 'AnnotationEditTool', 'FormDesignerEditTool', toolItem4,
'CommentTool', 'SubmitForm']}
pdfviewer.appendTo('#PdfViewer');
//To handle custom toolbar click event.
pdfviewer.toolbarClick = function (args) {

```

```
if (args.item && args.item.id === 'print') {  
  pdfviewer.print.print();  
}  
else if (args.item && args.item.id === 'download') {  
  pdfviewer.download();  
}  
};  
function OnCreateSearch(this: any): any {  
  this.addIcon('prepend', 'e-icons e-search');  
}
```

Note: Default value of toolbar items is ['OpenOption', 'PageNavigationTool', 'MagnificationTool', 'PanTool', 'SelectionTool', 'SearchOption', 'PrintOption', 'DownloadOption', 'UndoRedoTool', 'AnnotationEditTool', 'FormDesignerEditTool', 'CommentTool', 'SubmitForm']

Align Property

The align property is used to specify the alignment of a toolbar item within the toolbar.

Left: Aligns the item to the left side of the toolbar.

Right: Aligns the item to the right side of the toolbar.

Tooltip Property

The tooltip property is used to set the tooltip text for a toolbar item. Tooltip provides additional information when a user hovers over the item.

CssClass Property

The cssClass property is used to apply custom CSS classes to a toolbar item. It allows custom styling of the toolbar item.

Prefix Property

The prefix property is used to set the CSS class or icon that should be added as a prefix to the existing content of the toolbar item.

ID Property

The id property within a CustomToolbarItemModel is a compulsory attribute that plays a vital role in toolbar customization. It serves as a unique identifier for each toolbar item, facilitating distinct references and interactions.

When defining or customizing toolbar items, it is mandatory to assign a specific and descriptive id to each item.

These properties are commonly used when defining custom toolbar items with the CustomToolbarItemModel in the context of Syncfusion PDF Viewer. When configuring the toolbar using the ToolbarSettings property, you can include these properties to customize the appearance and behavior of each toolbar item.

Note: When customizing toolbar items, you have the flexibility to include either icons or text based on your design preference.

[View sample in GitHub](#)

Troubleshooting

Document Loading Issues in Version 23.1 or Newer

If you're experiencing problems with your document not rendering in the viewer, especially when using version 23.1 or a newer version, follow these troubleshooting steps to resolve the issue:

1. **Check for viewer.dataBind() Requirement:** Ensure that you have called `viewer.dataBind()` as required in version 23.1 or newer. This explicit call is essential for initializing data binding and document rendering correctly. It is must to call the `dataBind()` method before load.

```
`javascript
```

```
var viewer = new ej.pdfviewer.PdfViewer ({
  serviceUrl: 'https://services.syncfusion.com/js/production/api/pdfviewer'});

ej.pdfviewer.PdfViewer.Inject(ej.pdfviewer.Toolbar, ej.pdfviewer.Magnification,
ej.pdfviewer.BookmarkView, ej.pdfviewer.ThumbnailView, ej.pdfviewer.TextSelection,
ej.pdfviewer.TextSearch, ej.pdfviewer.Print, ej.pdfviewer.Navigation, ej.pdfviewer.LinkAnnotation,
ej.pdfviewer.Annotation, ej.pdfviewer.FormFields, ej.pdfviewer.FormDesigner);

viewer.appendTo('#pdfViewer');

viewer.dataBind();

viewer.load('https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf', null);
`
```

2. **Verify Document Source:** Confirm that the document source or URL you're trying to display is valid and accessible. Incorrect URLs or document paths can lead to loading issues.
3. **Network Connectivity:** Ensure that your application has a stable network connection. Document rendering may fail if the viewer can't fetch the document due to network issues.
4. **Console Errors:** Use your browser's developer tools to check for any error messages or warnings in the console. These messages can provide insights into what's causing the document not to load.
5. **Loading Sequence:** Make sure that you're calling `viewer.dataBind()` and initiating document loading in the correct sequence. The viewer should be properly initialized before attempting to load a document.
6. **Update Viewer:** Ensure that you're using the latest version of the viewer library or framework. Sometimes, issues related to document loading are resolved in newer releases.
7. **Cross-Origin Resource Sharing (CORS):** If you're loading documents from a different domain, ensure that CORS headers are correctly configured to allow cross-origin requests.
8. **Content Security Policies (CSP):** Check if your application's Content Security Policy allows the loading of external resources, as this can affect document loading. Refer [here](#) to troubleshoot.

By following these troubleshooting steps, you should be able to address issues related to document loading in version 23.1 or newer, ensuring that your documents render correctly in the viewer.

Pivot Table

Getting started in EJ2 JavaScript Pivotview control

This section explains the steps to create a simple Essential JS 2 **Pivot Table** and demonstrates the basic usage of the [pivot table](#) control in a JavaScript application.

Dependencies

The following list of dependencies are required to use the pivot table component in your application.

```
`javascript
|-- @syncfusion/ej2-pivotview
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-excel-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-calendars
|-- @syncfusion/ej2-charts
|-- @syncfusion/ej2-svg-base
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-grids
`,`
```

Setup for local environment

Refer the following steps for setup your local environment.

Step 1: Create a root folder `myapp` for your application.

Step 2: Create `myapp/resources` folder to store local scripts and styles files.

Step 3: Create `myapp/index.js` and `myapp/index.html` files for initializing Essential JS 2 pivot table control.

Adding Syncfusion resources

The Essential JS 2 pivot table control can be initialized by using either of the following ways.

- Using local script and style.
- Using CDN link for script and style.

Using local script and style

You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

After installing the Essential JS 2 product build, you can copy the pivot table scripts and style file into the `resources/scripts` and `resources/styles` folder.

Refer the below code to find location pivot table's script and style file.

Syntax:

Script: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js**

Styles: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css**

Example:

Script: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-pivotview/dist/global/ej2-pivotview.min.js**

Styles: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-pivotview/styles/material.css**

After copying the files, then you can refer the pivot table and its [dependency](#) scripts and styles into the **index.html** file.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 pivot table's dependent material theme -->
<link href="resources/ej2-base/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-buttons/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-calendars/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-dropdowns/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-inputs/styles/material.css" rel="stylesheet" type="text/css"/>
```

```

<link href="resources/ej2-lists/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-popups/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-navigations/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-grids/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 pivot table's material theme -->
<link href="resources/ej2-pivotview/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 pivot table's dependent scripts -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-buttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-file-utils.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-compression.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-excel-export.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-lists.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-pdf-export.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-svg-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-inputs.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-calendars.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-dropdowns.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-navigations.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-charts.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-grids.min.js" type="text/javascript"></script>
<!-- Essential JS 2 pivot table's global script -->
<script src="resources/scripts/ej2-pivotview.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>

```

In this illustration, we have referred **material** theme. You can also refer other themes like bootstrap, fabric, high-contrast etc.,

Using CDN link for script and style

Using CDN link, you can directly refer the pivot table control's script and style into the `index.html`.

Refer the pivot table's CDN links as below

Syntax:

Script: <http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js>

Styles: http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css

Example:

Script: <http://cdn.syncfusion.com/ej2/ej2-pivotview/dist/global/ej2-pivotview.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-pivotview/styles/material.css>

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 pivot table's dependent material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-calendars/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 pivot table's material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-pivotview/styles/material.css" rel="stylesheet"
type="text/css"/>
```

```
<!-- Essential JS 2 Grid's dependent scripts -->
<script src="http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-buttons/dist/global/ej2-buttons.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-file-utils/dist/global/ej2-file-utils.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-compression/dist/global/ej2-compression.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-excel-export/dist/global/ej2-excel-export.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-lists/dist/global/ej2-lists.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-pdf-export/dist/global/ej2-pdf-export.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-popups.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-svg-base/dist/global/ej2-svg-base.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-inputs/dist/global/ej2-inputs.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-calendars/dist/global/ej2-calendars.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-dropdowns/dist/global/ej2-dropdowns.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-charts/dist/global/ej2-charts.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js"
type="text/javascript"></script>
<!-- Essential JS 2 pivot table's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-pivotview/dist/global/ej2-pivotview.min.js"
type="text/javascript"></script>
</head>
<body>
```

```
</body>
```

```
</html>
```

```
,
```

```
<!-- markdownlint-disable MD028 -->
```

In this illustration, we have referred **material** theme. You can also refer other themes like bootstrap, fabric, high-contrast etc.,

Browser compatibility

Polyfills are required to use the Pivot Table in Internet Explorer 11 browser. Refer the [documentation](#) for more details.

Initializing pivot table component in the application

Pivot Table component can be initialized using the following code. To get started, add the pivot table component in **index.js** and **index.html** files using the following code.

To get started, add **div** element for pivot table control in **index.html**. Then refer the **index.js** file into the **index.html** file.

```
`html
```

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
```

```
<title>Essential JS 2</title>
```

```
<!-- Essential JS 2 pivot table's dependent material theme -->
```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet" type="text/css"/>
```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet" type="text/css"/>
```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-calendars/styles/material.css" rel="stylesheet" type="text/css"/>
```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css" rel="stylesheet" type="text/css"/>
```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet" type="text/css"/>
```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet" type="text/css"/>
```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet" type="text/css"/>
```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet" type="text/css"/>
```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet" type="text/css"/>
```

```
<!-- Essential JS 2 pivot table's material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-pivotview/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Pivot table's dependent scripts -->
<script src="http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-buttons/dist/global/ej2-buttons.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-file-utils/dist/global/ej2-file-utils.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-compression/dist/global/ej2-compression.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-excel-export/dist/global/ej2-excel-export.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-lists/dist/global/ej2-lists.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-pdf-export/dist/global/ej2-pdf-export.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-popups.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-svg-base/dist/global/ej2-svg-base.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-inputs/dist/global/ej2-inputs.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-calendars/dist/global/ej2-calendars.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-dropdowns/dist/global/ej2-dropdowns.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-charts/dist/global/ej2-charts.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js"
type="text/javascript"></script>
<!-- Essential JS 2 pivot table's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-pivotview/dist/global/ej2-pivotview.min.js"
type="text/javascript"></script>
```



```

</head>
<body>
<!-- Add the HTML <div> element for pivot table -->
<div id="PivotTable"></div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>
`

```

Next we need to initialize pivot table component using **`ej.pivotview.PivotView()`** instance as follows.

```

`javascript
var pivotTableObj = new ej.pivotview.PivotView();
`

```

Assigning sample data to pivot table component

The sample data is assigned to the pivot table component through `dataSource` property under `dataSourceSettings`.

Place the following pivot table code in the `index.js`.

```

`javascript
var pivotData = [
{ 'Sold': 31, 'Amount': 52824, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q1' },
{ 'Sold': 51, 'Amount': 86904, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q2' },
{ 'Sold': 90, 'Amount': 153360, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q3' },
{ 'Sold': 25, 'Amount': 42600, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q4' },
{ 'Sold': 27, 'Amount': 46008, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2016', 'Quarter': 'Q1' }];
var pivotTableObj = new ej.pivotview.PivotView({
dataSourceSettings: {
dataSource: pivotData,
}
});
pivotTableObj.appendTo('#PivotTable');
`

```

Adding fields to row, column, values and filters axes

Pivot Table component initialized and sample data assigned. Now add the fields to row, column, values and filters axes using following code.

- **rows**: Collection of fields that needs to be displayed in row axis of pivot table.
- **columns**: Collection of fields that needs to be displayed in column axis of pivot table.
- **values**: Collection of fields that needs to be displayed as aggregated numeric values in pivot table.
- **filters**: Filter the values in other axis based on the collection of filter fields in pivot table.

In-order to define each field in the respective axis, the following basic properties should be set.

- **name**: It allows to set the field name from the bound data source. It's casing should match exactly like in the data source and if not set properly, the pivot table will not be rendered.
- **caption**: It allows to set the field caption, which is the alias name of the field that needs to be displayed in the pivot table.
- **type**: It allows to set the summary type of the field. By default, SummaryType Sum is applied.

Place the following pivot table code in the `index.js`.

```
`javascript
var pivotData = [
  { 'Sold': 31, 'Amount': 52824, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q1' },
  { 'Sold': 51, 'Amount': 86904, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q2' },
  { 'Sold': 90, 'Amount': 153360, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q3' },
  { 'Sold': 25, 'Amount': 42600, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q4' },
  { 'Sold': 27, 'Amount': 46008, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2016', 'Quarter': 'Q1' }];
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name: 'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount', caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }]
  }
});
```

```
pivotTableObj.appendTo('#PivotTable');
`
```

Apply formatting to value fields

Formatting defines a way in which values should be displayed. For example, format “C” denotes the values should be displayed in currency pattern. To do so, define the name and format properties to formatSettings tag. In this illustration, the name property is set as Amount, a field from value section and its format is set as currency. Likewise, we can set format for other value fields as well and add it to formatSettings tag.

Only fields from value axis, which is in the form of numeric data values are applicable for formatting.

Place the following pivot table code in the `index.js`.

```
`javascript
var pivotData = [
  { 'Sold': 31, 'Amount': 52824, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q1' },
  { 'Sold': 51, 'Amount': 86904, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q2' },
  { 'Sold': 90, 'Amount': 153360, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q3' },
  { 'Sold': 25, 'Amount': 42600, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q4' },
  { 'Sold': 27, 'Amount': 46008, 'Country': 'France', 'Products': 'Mountain Bikes', 'Year': 'FY 2016', 'Quarter': 'Q1' }];
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name: 'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount', caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: [],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
  }
});
pivotTableObj.appendTo('#PivotTable');
`
```

Enable Grouping Bar

The Grouping Bar feature automatically populates fields from the bound data source and allows end users to drag fields between different axes such as columns, rows, values, and filters, and create pivot table at runtime. It can be enabled by setting the [showGroupingBar](#) property to **true**. To know more about grouping bar, [refer](#) here.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    allowLabelFilter: true,
    allowValueFilter: true,
    enableSorting: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  showGroupingBar: true
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable Pivot Field List

The component provides a built-in Field List similar to Microsoft Excel. It allows you to add or remove fields and also rearrange the fields between different axes, including column, row, value, and filter along with filter and sort options dynamically at runtime. It can be enabled by setting the [showFieldList](#) property to **true**.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    showFieldList: true

```

```
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Calculated field

The calculated field feature allows user to insert or add a new calculated field based on the available fields from the bound data source. It can be customized using the [calculatedFieldsSettings](#) property through code behind. The setting required for calculate field feature at code behind are:

- [name](#): it allows to indicate the given calculated field with unique name.
- [formula](#): it allows to set the formula base on the given data source.

Also calculated fields can be added at run time through the built-in dialog. The dialog can be enabled by setting the [allowCalculatedField](#) property to **true**.

Calculated field is applicable only for value fields.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }, { name: 'Total', caption: 'Total Amount', type:
'CalculatedField' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }, { name: 'Total',
format: 'C2' }],
    filters: [],
    calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Sold)"' }]
  },
  showFieldList: true,
  height: 350,
  allowCalculatedField: true
});
pivotTableObj.appendTo('#PivotTable');
var btn = new ej.buttons.Button({ isPrimary: true });
btn.appendTo('#CalculatedField');
document.getElementById('CalculatedField').addEventListener('click', () => {
  pivotTableObj.calculatedFieldModule.createCalculatedFieldDialog();
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="CalculatedField">Calculated Field</div>
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exploring Filter axis

The filter axis contains collection of fields that would act as master filter over the data bound in row, column and value axes of the pivot table. The fields along with filter members could be set to filter axis either through report via code behind or by dragging and dropping fields from other axes to filter axis via grouping bar or field list at runtime.

Place the following pivot table code in the `index.js`.

INDEX.JS


```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Products' }],
    filters:[{ name: 'Country' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }]
  },
  height: 350,
  showFieldList: true,
  showGroupingBar:true
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Run the application

Now, run the `index.html` in web browser, it will render the **Essential JS 2 pivot table** control.

Output will be displayed as follows.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Sold) "' }],
    },
    height: 350,
    showFieldList: true,
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Deploy the Application

The Essential JS 2 pivot table control features are segregated into individual feature-wise modules. The [Essential Studio JavaScript \(Essential JS 2\)](#) build and [CDN](#) scripts contains code for all features used in pivot table and hence we suggest to not to use them in production. We strongly recommend you to generate script files to use in production using our [Custom Resource Generator\(CRG\)](#) for Essential JS 2. CRG will allow you to generate the bundled script for the currently enabled features in pivot table.

You can also explore our [JavaScript Pivot Table example](#) that shows how to rendering of the pivot table with drill-up and drill-down functionality bound to a relational report.

Data binding in EJ2 JavaScript Pivotview control

JSON

For JSON data binding, the `type` property under [dataSourceSettings](#) needs to be set as `JSON`. By default, the default value is assumed as `JSON`.

Binding JSON data via local

In-order to bind local JSON data to the pivot table user can assign the local variable holding the JSON data to the [dataSource](#) property under [dataSourceSettings](#).

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: []
  },
  height: 350,
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using local variable, the JSON data can also be bound to the pivot table using [DataManager](#) option with the help of [JsonAdaptor](#). Here the instance of [DataManager](#) holding JSON data is assigned to [dataSource](#) property under [dataSourceSettings](#). The use of [DataManager](#) is optional here.

INDEX.JS

```

var data = new ej.data.DataManager({ json: pivotData, adaptor: new
ej.data.JsonAdaptor });
var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: data,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: [],
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

In the meantime, the JSON data from the local *.json file type can also be connected to the pivot table via the file uploader option. Here, the resulting string after uploading the file needs to be converted to JSON data that can be assigned to the [dataSource](#) property under [dataSourceSettings](#). The following code example illustrates the same.

```
`javascript
import { PivotView } from '@syncfusion/ej2-pivotview';
import { Uploader } from '@syncfusion/ej2-inputs';
// Step 1: Initiate the file uploader
let uploadObj: Uploader = new Uploader({
});
uploadObj.appendTo('#fileupload');
let input = document.querySelector('input[type="file"]');
// Step 2: Add the event listener which fires when the *.JSON file is uploaded.
input.addEventListener('change', function (e: Event) {
// Step 3: Initiate the file reader
let reader: FileReader = new FileReader();
reader.onload = function () {
// Step 4: Getting the string output which is to be parsed as JSON.
let result: any = JSON.parse(reader.result as string);
let pivotGridObj: PivotView = new PivotView({
dataSourceSettings: {
// Step 5: The JSON result to be bound as data source.
dataSource: result
// Step 6: The appropriate report needs to be provided here.
},
});
pivotGridObj.appendTo('#PivotTable');
};
reader.readAsText((input as any).files[0]);
});
`
```

Binding JSON data via remote

In-order to bind remote JSON data, mention the endpoint [URL](#) under [dataSourceSettings](#) property. The [URL](#) property supports both direct downloadable file (*.json) and web service URL.

INDEX.JS

```
var pivotGridObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    url: 'https://cdn.syncfusion.com/data/sales-analysis.json',
    expandAll: false,
    rows: [
      { name: 'EnerType', caption: 'Energy Type' }
    ],
    columns: [
      { name: 'EneSource', caption: 'Energy Source' }
    ],
    values: [
      { name: 'PowUnits', caption: 'Units (GWh)' },
      { name: 'ProCost', caption: 'Cost (MM)' }
    ],
    filters: []
  }
});
pivotGridObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```



```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

CSV

For CSV data binding, the **type** property under [dataSourceSettings](#) needs to be set as **CSV** mandatorily.

The CSV format is considered to be the most compact format compared to JSON since it is half the size of JSON. This helps to reduce the bandwidth while transferring to the browser.

Binding CSV data via local

In-order to bind local CSV data to the pivot table, user needs to convert it as string array and then directly assign it to the [dataSource](#) property under [dataSourceSettings](#).

INDEX.JS

```

var pivotObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: getCSVDData(),
        type: 'CSV',
        expandAll: false,
        formatSettings: [{ name: 'Total Cost', format: 'C0' }, { name: 'Total Revenue', format: 'C0' }, { name: 'Total Profit', format: 'C0' }],
        drilledMembers: [{ name: 'Item Type', items: ['Baby Food'] }],
        rows: [
            { name: 'Country' },
            { name: 'Region' }
        ],
        columns: [
            { name: 'Sales Channel' },
            { name: 'Item Type' }
        ],
        values: [
            { name: 'Total Profit' },
            { name: 'Total Cost' },
            { name: 'Total Revenue' }
        ],
        filters: []
    },
    height: 290,

```

```

        width: '100%'
    });
    pivotObj.appendTo('#PivotTable');
    function getCSVData() {
        var dataSource = [];
        var jsonObject = csvdata.split(/\r?\n|\r/);
        for (var i = 0; i < jsonObject.length; i++) {
            dataSource.push(jsonObject[i].split(','));
        }
        return dataSource;
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In the meantime, the CSV data from the local *.csv file type can also be connected to the pivot table via the file uploader option. Here, the resulting string after uploading the file needs to be converted to string array that can be assigned to the [dataSource](#) property under [dataSourceSettings](#). The following code example illustrates the same.

```

`javascript
import { PivotView } from '@syncfusion/ej2-pivotview';
import { Uploader } from '@syncfusion/ej2-inputs';
// Step 1: Initiate the file uploader
let uploadObj: Uploader = new Uploader({
});
uploadObj.appendTo('#fileupload');
// Step 2: Add the event listener which fires when the *.CSV file is uploaded.
let input = document.querySelector('input[type="file"]');
input.addEventListener('change', function (e: Event) {
// Step 3: Initiate the file reader
let reader: FileReader = new FileReader();
reader.onload = function () {
// Step 4: Getting the string output which is to be converted as string[][].
let result: string[][] = (reader.result as string).split('\n').map(function (line) {
return line.split(',');
});
let pivotGridObj: PivotView = new PivotView({
dataSourceSettings: {
// Step 5: The string[][] result to be bound as data source.
dataSource: result,
type: 'CSV',

```

// Step 6: The appropriate report needs to be provided here.

```

},
});
pivotGridObj.appendTo('#PivotTable');
};
reader.readAsText((input as any).files[0]);
});
`

```

Binding CSV data via remote

In-order to bind remote CSV data, mention the endpoint [URL](#) under [dataSourceSettings](#) property. The [URL](#) property supports both direct downloadable file (*.csv) and web service URL.

INDEX.JS

```

var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    url: 'https://bi.syncfusion.com/productservice/api/sales',
    type: 'CSV',
    expandAll: false,
    enableSorting: true,
    formatSettings: [{ name: 'Total Cost', format: 'C0' }, { name:
'Total Revenue', format: 'C0' }, { name: 'Total Profit', format: 'C0' }],
    drilledMembers: [{ name: 'Item Type', items: ['Baby Food' ]}],
    rows: [
      { name: 'Region' },
      { name: 'Country' }
    ],
    columns: [
      { name: 'Item Type' },
      { name: 'Sales Channel' }
    ],
    values: [
      { name: 'Total Cost' },
      { name: 'Total Revenue' },
      { name: 'Total Profit' }
    ],
    filters: []
  },
  height: 300,
  width: '100%'
});
pivotObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Remote Data Binding

To interact with remote data source, provide the endpoint [URL](#) within [DataManager](#) along with appropriate [adaptor](#). By default, [DataManager](#) uses [ODataAdaptor](#) for remote data-binding.

INDEX.JS

```
var data = new ej.data.DataManager({
```

```

url: 'https://bi.syncfusion.com/northwindservice/api/orders',
adaptor: new ej.data.WebApiAdaptor,
crossDomain: true
});
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: [],
    expandAll: true,
    filters: [],
    columns: [{ name: 'ProductName', caption: 'Product Name' }],
    rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
    formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
    values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
  },
  height: 350,
  width: '100%',
  gridSettings: { columnWidth: 120 }
});
pivotTableObj.appendTo('#PivotTable');
data.executeQuery(new ej.data.Query().take(8)).then(function (e) {
  pivotTableObj.dataSourceSettings.dataSource = e.result;
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

```

```

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Binding with OData services

OData is a standardized protocol for creating and consuming data. User can retrieve data from OData service using the [DataManager](#). Refer to the following code example for remote data binding using OData service.

INDEX.JS

```

var data = new ej.data.DataManager({
    url:
    'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders/',
    adaptor: new ej.data.ODataAdaptor,
    crossDomain: true
});
var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: data,
        expandAll: true,
        filters: [],
        columns: [{ name: 'CustomerID', caption: 'Customer Name' }],
        rows: [{ name: 'OrderID', caption: 'Order ID' }],
        values: [{ name: 'Freight' }]
    },
    height: 350,
    width: '100%',
    gridSettings: { columnWidth: 120 }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```


Binding with OData V4 services

The OData V4 is an improved version of OData protocols, and the [DataManager](#) can be used to retrieve and consume OData V4 services. For more details on OData V4 services, refer to the [OData documentation](#). To bind OData V4 service, use the [ODataV4Adaptor](#).

INDEX.JS

```
var data = new ej.data.DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/',
  adaptor: new ej.data.ODataV4Adaptor,
  crossDomain: true
});
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: data,
    expandAll: true,
    filters: [],
    columns: [{ name: 'CustomerID', caption: 'Customer Name' }],
    rows: [{ name: 'OrderID', caption: 'Order ID' }],
    values: [{ name: 'Freight' }]
  },
  height: 350,
  width: '100%',
  gridSettings: { columnWidth: 120 }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Web API

User can use [WebApiAdaptor](#) to bind pivot table with Web API created using OData endpoint.

`ts

```

import { PivotView, IDataset } from '@syncfusion/ej2-pivotview';
import { DataManager, WebApiAdaptor, Query, ReturnOption } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://bi.syncfusion.com/northwindservice/api/orders',
    adaptor: new WebApiAdaptor,
    crossDomain: true
});
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: data,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name: 'ShipCity', caption: 'Ship City' }],

```

```

formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit Price' }]
},
height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

Querying in Data Manager

By default, the data manager retrieves all the data from the provider which is mapped in it. The data from the provider can be filtered, sorted, paged, etc. by setting the own query in [defaultQuery](#) property in the data manager instance.

INDEX.JS

```

var remoteData = new ej.data.DataManager({
  url:
  'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders',
  adaptor: new ej.data.ODataAdaptor(),
  crossDomain: true
});
remoteData.defaultQuery = new ej.data.Query().take(2);
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: remoteData,
    expandAll: false,
    columns: [{ name: 'CustomerID', caption: 'Customer ID' }],
    rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
    values: [{ name: 'Freight' }]
  },
  height: 350,
  width: '100%',
  gridSettings: { columnWidth: 120 }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Mapping

One can define field information like alias name (caption), data type, aggregation type, show and hide subtotals etc. using the [fieldMapping](#) property under [dataSourceSettings](#). The available options are,

- [name](#) - It is to specify the appropriate field name.
- [caption](#) - It is to set the alias name (caption) to the specific field. Instead of actual field name, the alias name (caption) will be set in the UI of the pivot table.
- [type](#) - It is to display values in the pivot table with appropriate aggregation such as sum, product, count, average, minimum, maximum, etc. Its default value is **sum**. This option is applicable only for relational data source.
- [axis](#) - It will help to display the field in specified axis such as row/column/value/filter axis of the pivot table.

- [showNoDataItems](#) - It is to show all the members of a specific field to the pivot table, even if there are no data in the intersection of the row and column. The default value is **false**. This option is applicable only for relational data source.
- [baseField](#) - For the aggregate types like "DifferenceFrom" or "PercentageOfDifferenceFrom" or "PercentageOfParentTotal", selective field is assigned for comparison via this property.
- [baseItem](#) - For the aggregate types like "DifferenceFrom" or "PercentageOfDifferenceFrom" or "PercentageOfParentTotal", selective member in a field is assigned for comparison via this property.
- [expandAll](#) - It is to expand or collapse all headers of a specific field in row and column axes of the pivot table. The default value is **false**.
- [showSubTotals](#) - It is to show or hide sub-totals of a specific field in row and column axis of the pivot table. The default value is **true**.
- [isNamedSet](#) - It is to set whether the specified field is named set or not. In general, the named set is a set of dimension members or a set expression (MDX query) to be created as a dimension in the SSAS OLAP cube itself. The default value is **false** and this option is applicable only for OLAP data source.
- [isCalculatedField](#) - It is to set whether the specified field is a calculated field or not. In general, a calculated field is created from the bound data source or using simple formula with basic arithmetic operators in the pivot table. The default value is **false** and this option is applicable only for OLAP data source.
- [showFilterIcon](#) - It is to show or hide the filter icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This filter icon is used to filter the members of a specified field at runtime in the pivot table. The default value is **true**.
- [showSortIcon](#) - It is to show or hide the sort icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This sort icon is used to order members of a specified field either in ascending or descending at runtime. The default value is **true**.
- [showRemoveIcon](#) - It is to show or hide the remove icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This remove icon is used to remove the specified field during runtime. The default value is **true**.
- [showValueTypeIcon](#) - It is to show or hide the value type icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This value type icon helps to select the appropriate aggregation type to specified value field at runtime. The default value is **true**.
- [showEditIcon](#) - It is to show or hide the edit icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This edit icon is used to modify caption, formula, and format of a specified calculated field at runtime. The default value is **true**.
- [allowDragAndDrop](#) - It is to restrict specific field's button from being dragged on runtime in the grouping bar and field list UI. This will prevent from altering the current report. The default value is **true**.
- [dataType](#) - It is to specify the type of the field like 'string', 'number', 'datetime', 'date', and 'boolean'.
- [groupName](#) - It is to display fields in the field list UI by grouping them under the desired folder name.

The main purpose of these mapping options is to configure each field that is not part of the initial pivot report. Even if any field that is part of this mapping is defined here, the value set in the initial report will have the highest preceding.

This option is applicable only for relational data source.

In the below code sample, visibility of the field button icons are configured.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    allowLabelFilter: true,
    allowValueFilter: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: [],
    fieldMapping: [
      { name: 'Year', showFilterIcon: false },
      { name: 'Quarter', showSortIcon: false },
      { name: 'Country', allowDragAndDrop: false },
      { name: 'Products', showFilterIcon: false, showRemoveIcon:
false },
      { name: 'Sold', showValueTypeIcon: false },
      { name: 'Amount', showValueTypeIcon: false },
    ]
  },
  height: 350,
  showGroupingBar: true,
  showFieldList: true,
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Values in row axis

By default, the value fields are plotted in column axis. To plot those fields in row axis, use the [valueAxis](#) property by setting its value as **row**. By default, it holds the value **column**.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: [],
        valueAxis: 'row'
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```


Values at different positions

By default, the value fields are placed at the end of the row or column axis. To place those value fields in different positions, use the [valueIndex](#) property and set the value to an appropriate index position. Its default value is -1, which denotes the last position. The [valueIndex](#) property is dependent on the [valueAxis](#) property.

This support is only available for relational data sources. Also, enable the [showValuesButton](#) property in the [grouping bar](#) and [field list](#) UI to **true** to re-arrange the values fields at different positions via user interaction.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: [],
    valueIndex: 1
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show 'no data' items

By default, the pivot table only shows the field item if it has data in its row or column combination. To show all items that do not have data in row and column combination in the pivot table, use the [showNoDataItems](#) property by settings its value to true for the desired fields. In the following code sample, rows of the "Country" and "State" fields do not have data in all combination with "Date" column field.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: nodata,
        expandAll: true,
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        columns: [{ name: 'Date', showNoDataItems: true}],
        values: [{ name: 'Quantity', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
        rows: [{ name: 'Country', showNoDataItems: true }, { name: 'State',
showNoDataItems: true }],
        filters: []
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Always shows the value headers

To show the value header always in pivot table, even if it holds a single value, use the [alwaysShowValueHeader](#) property by settings its value as **true**.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: [],
    alwaysShowValueHeader: true
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize empty value cells

User can show custom string in empty value cells using the [emptyCellsTextContent](#) property in [dataSourceSettings](#) of the pivot table. Since the property is of string data type, user can fill empty value cells with any value like "0", "-", "*", "(blank)", etc. Its common for all value fields and can be configured through code behind.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: nodata,
        expandAll: true,
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        columns: [{ name: 'Date', showNoDataItems: true}],
        values: [{ name: 'Quantity', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
        rows: [{ name: 'Country', showNoDataItems: true }, { name: 'State',
showNoDataItems: true }],
        filters: [],
        emptyCellsTextContent: '**'
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Events

Load

The event [load](#) fires before initiate rendering of pivot table. It holds following parameters like **dataSourceSettings**, **fieldsType** and **pivotView**. In this event user can customize data source settings before initiating pivot table render module.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: nodata,
    expandAll: true,
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    columns: [{ name: 'Date', showNoDataItems: true }],
    values: [{ name: 'Quantity', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
    rows: [{ name: 'Country', showNoDataItems: true }, { name: 'State',
showNoDataItems: true }],
    filters: [],
    emptyCellsTextContent: '**'
  },
  height: 350,
  showGroupingBar: true,
  load: function (args) {
    args.dataSourceSettings.columns[0].caption = 'Order Date';
    args.dataSourceSettings.emptyCellsTextContent = '###'
  },
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

```

```

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

EnginePopulated

The event [enginePopulated](#) is triggered after engine is populated. It has following parameters - `dataSourceSettings`, `pivotFieldList` and `pivotValues`.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        allowLabelFilter: true,
        allowValueFilter: true,
        columns: [{ name: 'Year', caption: 'Production Year' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    showFieldList: true,
    enginePopulated: function (args) {
        args.dataSourceSettings.columns[0].caption = 'Total Population';
        args.dataSourceSettings.emptyCellsTextContent = '###'
    },
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML


```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

EnginePopulating

The event [enginePopulating](#) triggers before the pivot engine starts to populate and allows to customize the pivot datasource settings. It has following parameter `dataSourceSettings`.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    allowLabelFilter: true,
    allowValueFilter: true,
    columns: [{ name: 'Year', caption: 'Production Year' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  showFieldList: true,
  enginePopulated: function (args) {
    args.dataSourceSettings.columns[0].caption = 'Total Population';
    args.dataSourceSettings.emptyCellsTextContent = '###'
  },
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
popups/styles/material.css" rel="stylesheet">  
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
pivotview/styles/material.css" rel="stylesheet">  
  
<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-  
awesome.min.css" rel="stylesheet">  
  
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"  
type="text/javascript"></script>  
<script src="es5-datasource.js" type="text/javascript"></script>  
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type  
="text/javascript"></script>  
</head>  
<body>  
  
    <div id="container">  
        <div>  
            <div id="PivotTable"></div>  
        </div>  
    </div>  
<script>  
var ele = document.getElementById('container');  
if(ele) {  
    ele.style.visibility = "visible";  
}  
    </script>  
<script src="index.js" type="text/javascript"></script>  
</body></html>
```

See Also

- [Aggregation](#)
- [Show/Hide Totals](#)
- [Customize number, date, and time values](#)
- [Server Side Engine \(Optional\)](#)

Connecting To Data Source

MySQL in EJ2 JavaScript Pivotview control

This section describes how to retrieve data from a MySQL database using [MySqlClient](#) and bind it to the Pivot Table via a Web API controller.

Create a Web API service to fetch MySQL data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).

Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name

MyWebService

Location

C:\Users\username\source\repos

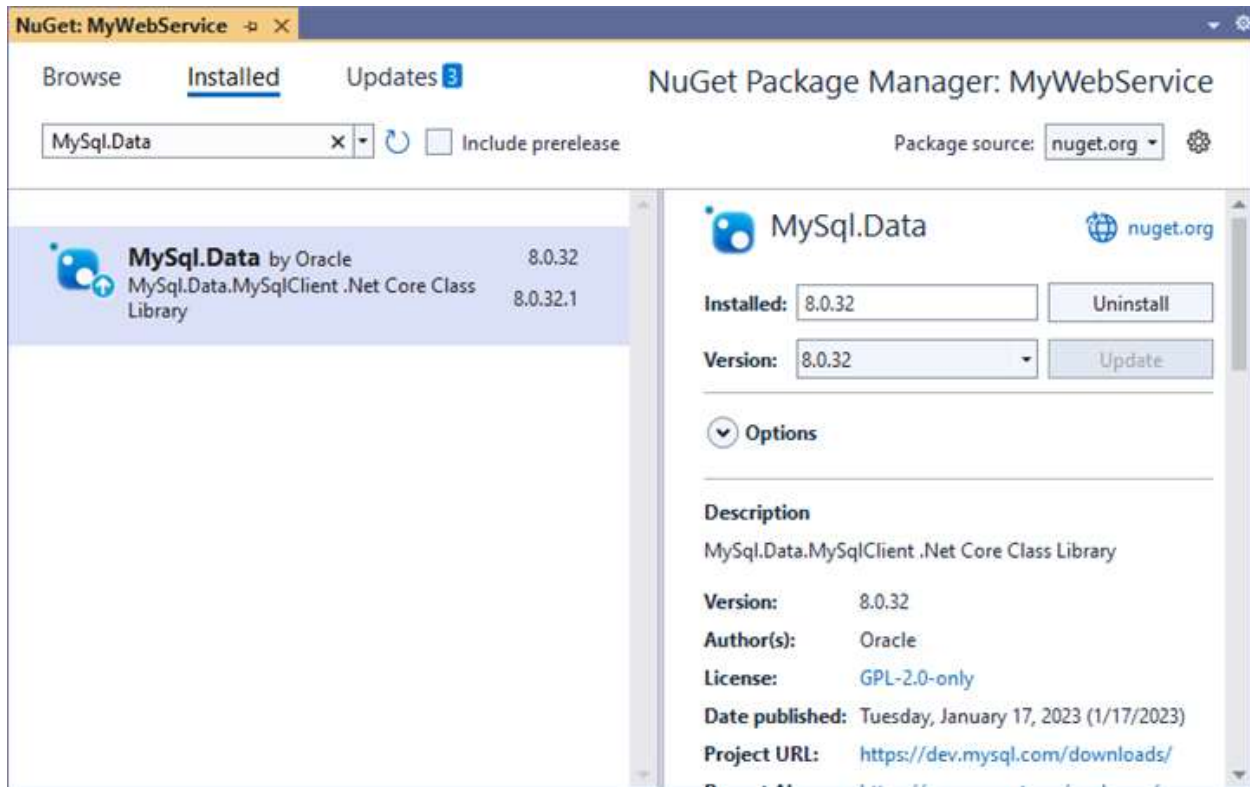
Solution name ⓘ

MyWebService

☐ Place solution and project in the same directory

Back Next

2. To connect a MySQL Server using the **MySqlClient** in our application, we need to install the [MySql.Data](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **MySql.Data** and install it.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **MySQLConnection** helps to connect the MySQL database. Next, using **MySQLCommand** and **MySQLDataAdapter** you can process the desired query string and retrieve data from the MySQL database. The **Fill** method of the **MySQLDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```
`c#
using Microsoft.AspNetCore.Mvc;
using MySQL.Data.MySqlClient;
using Newtonsoft.Json;
using System.Data;
namespace MyWebService.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class PivotController : ControllerBase
    {
        public dynamic GetMySQLResult()
        {
```

```
// Replace with your own connection string.
MySQLConnection connection = new MySqlConnection("<Enter your valid connection string here>");
connection.Open();
MySQLCommand command = new MySQLCommand("SELECT * FROM orders", connection);
MySQLDataAdapter dataAdapter = new MySQLDataAdapter(command);
DataTable dataTable = new DataTable();
dataAdapter.Fill(dataTable);
connection.Close();
return dataTable;
}
}
}
```

5. In the **Get()** method of the **PivotController.cs** file, the **GetMySQLResult** method is used to retrieve the MySQL data as a **DataTable**, which is then serialized into JSON string using **JsonConvert.SerializeObject()**.

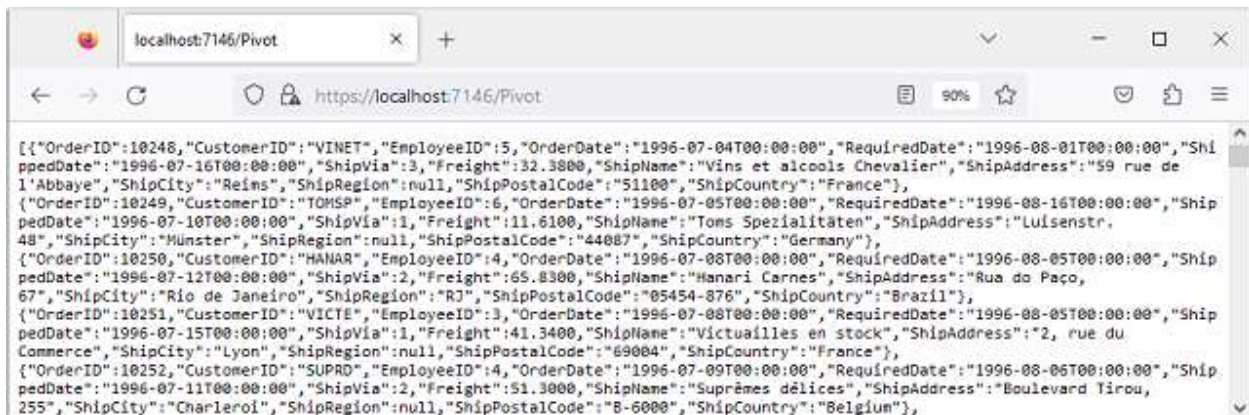
```
`c#
using Microsoft.AspNetCore.Mvc;
using MySql.Data.MySqlClient;
using Newtonsoft.Json;
using System.Data;
namespace MyWebService.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class PivotController : ControllerBase
    {
        [HttpGet(Name = "GetMySQLResult")]
        public object Get()
        {
            return JsonConvert.SerializeObject(GetMySQLResult());
        }
        public dynamic GetMySQLResult()
        {

```

```
// Replace with your own connection string.
MySQLConnection connection = new MySqlConnection("<Enter your valid connection string here>");
connection.Open();
MySQLCommand command = new MySqlCommand("SELECT * FROM orders", connection);
MySQLDataAdapter dataAdapter = new MySQLDataAdapter(command);
DataTable dataTable = new DataTable();
dataAdapter.Fill(dataTable);
connection.Close();
return dataTable;
}
}
}
```

6. Run the application and it will be hosted within the URL <https://localhost:7146>.

7. Finally, the retrieved data from MySQL database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:7146/Pivot>, as shown in the browser page below.



Connecting the Pivot Table to a MySQL database using the Web API service

1. Create a simple Javascript Pivot Table by following the **"Getting Started"** documentation [link](#).

2. Map the hosted Web API's URL link <https://localhost:7146/Pivot> to the Pivot Table in **index.js** by using the **url** property under **dataSourceSettings**.

```
`Javascript
var pivotObj = new ej.pivotview.PivotView({
dataSourceSettings: {
url: 'https://localhost:7146/Pivot',
//Other codes here...
```

```

},
});
pivotObj.appendTo('#PivotView');
`

```

3. Frame and set the report based on the data retrieved from the MySQL database.

```

`Javascript
var pivotObj = new ej.pivotview.PivotView({
dataSourceSettings: {
url: 'https://localhost:7146/Pivot',
columns: [{ name: 'ShipName' }],
values: [{ name: 'Freight', caption: 'Sum of Freight' }],
rows: [{ name: 'ShipCity' }],
},
showFieldList: true,
width: '100%',
height: 350,
});
pivotObj.appendTo('#PivotView');
`

```

When you run the sample, the resulting pivot table will look like this:

	Alfred's Futterkist	Alfreds Futterkist	Ana Trujillo Empa	Antonio Moreno	Around the Horn	B's
Aachen						
Albuquerque						
Anchorage						
Barcelona						
Barquisimeto						
Bergamo						
Berlin	196.12000000...	29.46				
Bern						

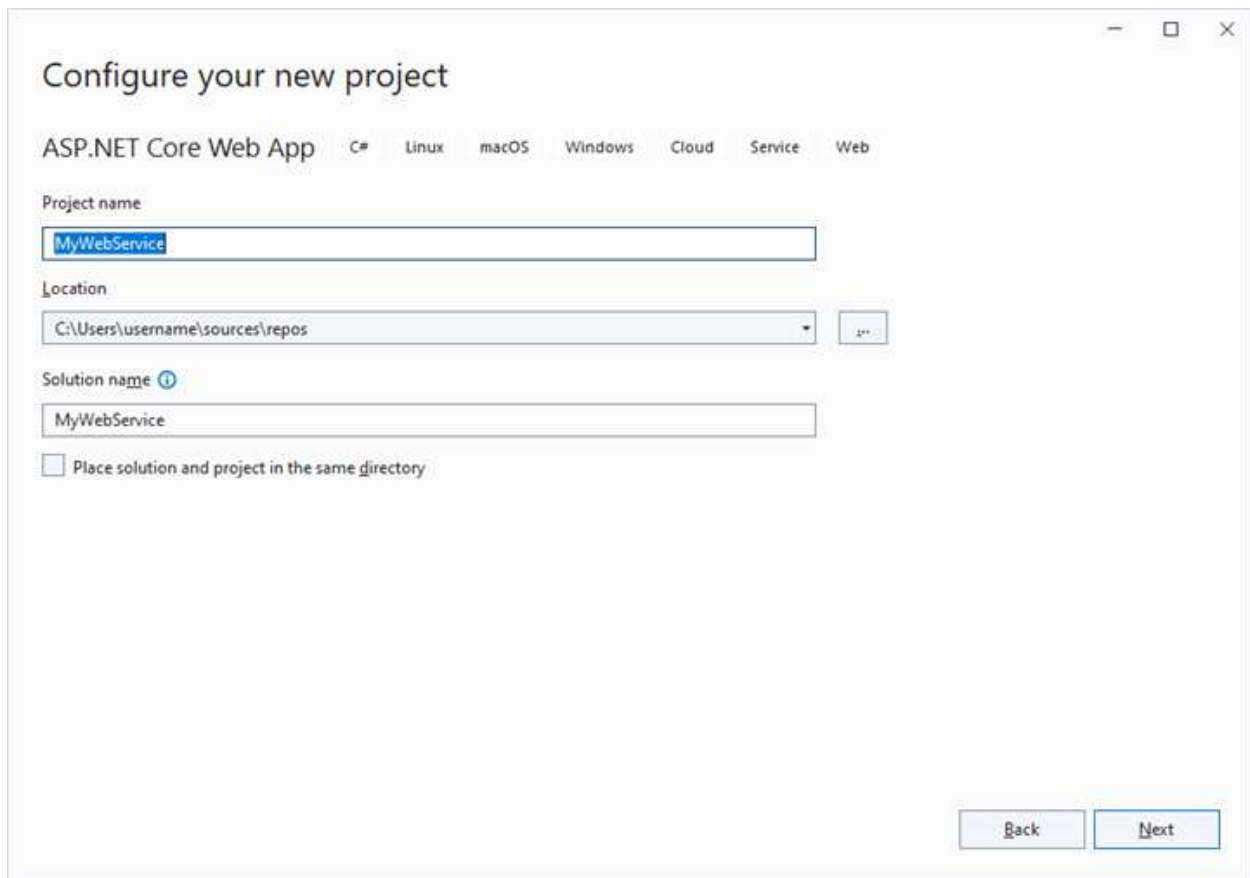
Explore our Javascript Pivot Table sample and ASP.NET Core Web Application to extract data from a MySQL database and bind to the Pivot Table in [this](#) GitHub repository.

Microsoft SQL server in EJ2 JavaScript Pivotview control

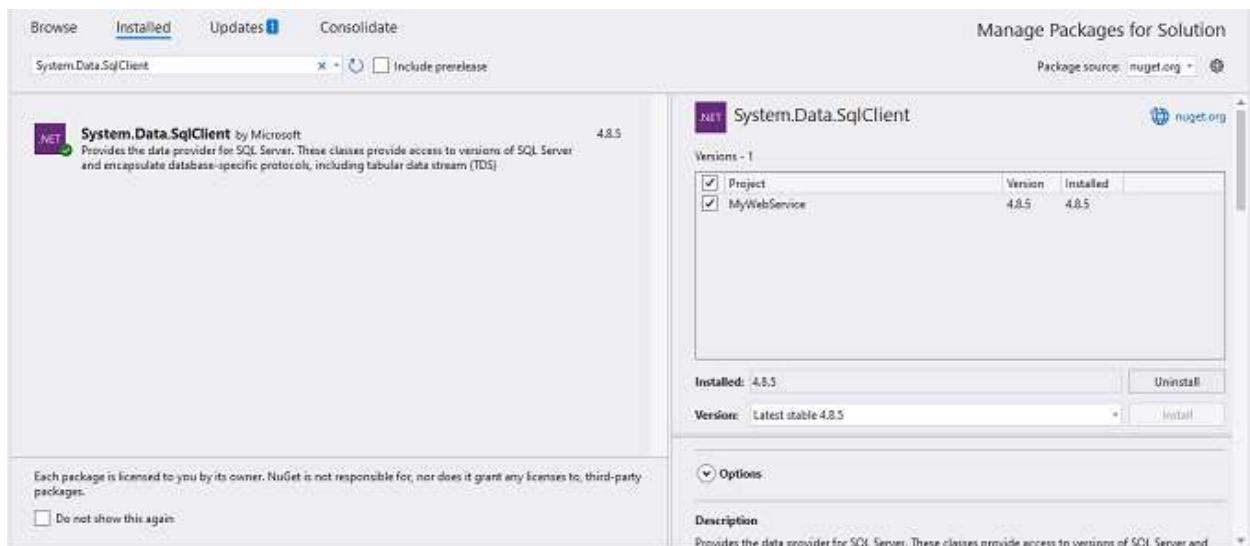
This section describes how to retrieve data from Microsoft SQL Server using [Microsoft SqlClient](#) and bind it to the Pivot Table via a Web API controller.

Create a Web API service to fetch SQL Server data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).



2. To connect a SQL Server using the **SqlClient** in our application, we need to install the [System.Data.SqlClient](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **System.Data.SqlClient** and install it.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **SqlConnection** helps to connect the SQL database (that is, Database1.mdf). Next, using **SqlCommand** and **SqlDataAdapter** you can process the desired SQL query string and retrieve data from the database. The **Fill** method of the DataAdapter is used to populate the SQL data into a **DataTable** as shown in the following code snippet.

```
`c#
using Microsoft.AspNetCore.Mvc;
using System.Data;
using System.Data.SqlClient;
namespace MyWebService.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class PivotController : ControllerBase
    {
        private static DataTable FetchSQLResult()
        {
            // Replace with your own connection string.
            string conSTR = @"<Enter your valid connection string here>";
            SqlConnection sqlConnection = new(conSTR);
            sqlConnection.Open();
            SqlCommand cmd = new(xquery, sqlConnection);
            SqlDataAdapter dataAdapter = new(cmd);
            DataTable dataTable = new();
            dataAdapter.Fill(dataTable);
            return dataTable;
        }
    }
}
```

5. In the **Get()** method of the **PivotController.cs** file, the **FetchSQLResult** method is used to retrieve the SQL data as a **DataTable**, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```
`c#
using Microsoft.AspNetCore.Mvc;
```

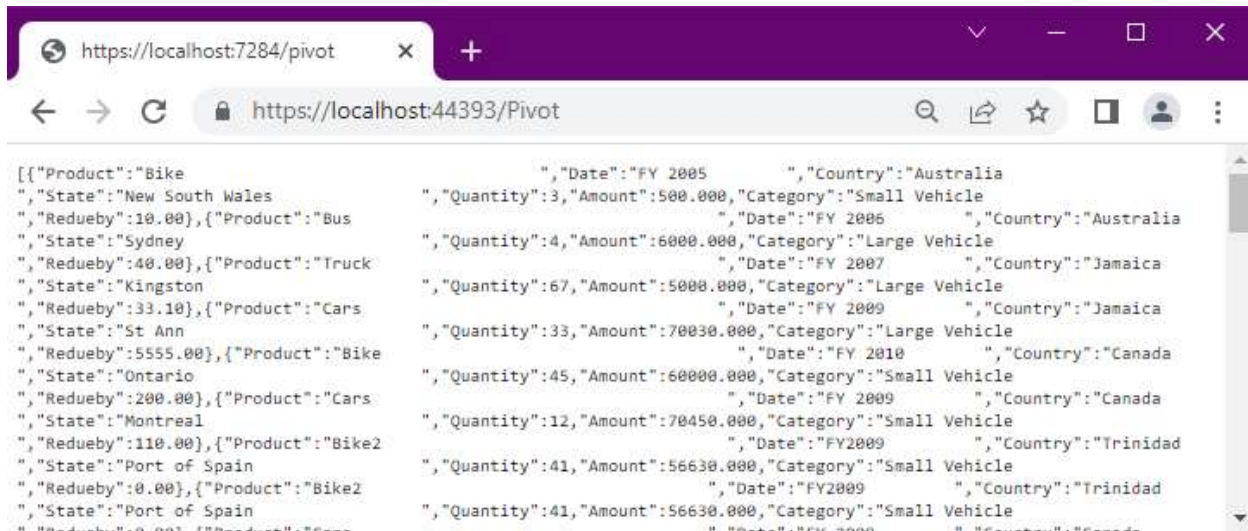
```

using Newtonsoft.Json;
using System.Data;
using System.Data.SqlClient;
namespace MyWebService.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class PivotController : ControllerBase
    {
        [HttpGet(Name = "GetSQLResult")]
        public object Get()
        {
            return JsonConvert.SerializeObject(FetchSQLResult());
        }
        private static DataTable FetchSQLResult()
        {
            string conSTR = @"<Enter your valid connection string here>";
            string xquery = "SELECT * FROM table1";
            SqlConnection sqlConnection = new(conSTR);
            sqlConnection.Open();
            SqlCommand cmd = new(xquery, sqlConnection);
            SqlDataAdapter dataAdapter = new(cmd);
            DataTable dataTable = new();
            dataAdapter.Fill(dataTable);
            return dataTable;
        }
    }
}

```

6. Run the web application and it will be hosted within the URL <https://localhost:44393>.

7. Finally, the retrieved data from SQL Server which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:44393/Pivot>, as shown in the browser page below.



Connecting the Pivot Table to a SQL database using the Web API service

1. Create a simple Javascript Pivot Table by following the **"Getting Started"** documentation [link](#).
2. Map the hosted Web API's URL link `https://localhost:44393/Pivot` to the Pivot Table in `index.js` by using the `url` property under `dataSourceSettings`.

```
`javascript
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    url: 'https://localhost:44393/Pivot',
    //Other codes here...
  }
});
pivotObj.appendTo('#PivotView');
```

3. Frame and set the report based on the data retrieved from the SQL database.

```
`javascript
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    url: 'https://localhost:44393/Pivot',
    expandAll: false,
    enableSorting: true,
    columns: [{ name: 'Product' }],
    values: [{ name: 'Quantity' }, { name: 'Amount', caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'State' }],
```

```

formatSettings: [{ name: 'Amount', format: 'C0' }],
filters: []
},
showFieldList: true,
width: '100%'
});
pivotObj.appendTo('#PivotView');
`

```

When you run the sample, the resulting pivot table will look like this:

	Large Vehicle		Small Vehicle		Grand Total	
	Quantity	Amount	Quantity	Amount	Quantity	Amount
▶ Australia	16	24000	12	2000	28	
▶ Canada			420	1649000	420	
▶ Jamaica	400	300120			400	
▶ Trinidad			328	453040	328	
Grand Total	416	324120	760	2104040	1176	

Explore our Javascript Pivot Table sample and ASP.NET Core Web Application to extract data from a SQL database and bind to the Pivot Table in [this](#) GitHub repository.

PostgreSQL in EJ2 JavaScript Pivotview control

This section describes how to consume data from PostgreSQL database using [Microsoft Npgsql](#) and bind it to the Pivot Table via a Web API controller.

Create a Web API service to fetch PostgreSQL data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).

Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name
MyWebService

Location
C:\Users\username\source\repos

Solution name
MyWebService

☐ Place solution and project in the same directory

Back Next

2. To connect a PostgreSQL Server using the **Npgsql** in our application, we need to install the [Npgsql.EntityFrameworkCore.PostgreSQL](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Npgsql.EntityFrameworkCore.PostgreSQL** and install it.

Browse Installed Updates Consolidate Manage Packages for Solution

Npgsql.EntityFrameworkCore.PostgreSQL x Include prerelease Package source: nuget.org

Npgsql.EntityFrameworkCore.PostgreSQL by Shay Rojansky, A 3.1.2
PostgreSQL/Npgsql provider for Entity Framework Core. 7.0.3

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
☐ Do not show this again

Npgsql.EntityFrameworkCore.PostgreSQL nuget.org

Versions - 1

Project	Version	Installed
MyWebService	3.1.2	3.1.2

Installed: 3.1.2 Uninstall

Version: 3.1.2 Install

Options

Description
PostgreSQL/Npgsql provider for Entity Framework Core.

3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **NpgsqlConnection** helps to connect the PostgreSQL database. Next, using **NpgsqlCommand** and **NpgsqlDataAdapter** you can process the desired PostgreSQL query string and retrieve data from the database. The **Fill** method of the **NpgsqlDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```
`c#  
using Microsoft.AspNetCore.Mvc;  
using Newtonsoft.Json;  
using System.Data;  
using Npgsql;  
namespace MyWebService.Controllers  
{  
    [ApiController]  
    [Route("[controller]")]  
    public class PivotController : ControllerBase  
    {  
        public dynamic GetPostgreSQLResult()  
        {  
            // Replace with your own connection string.  
            NpgsqlConnection connection = new NpgsqlConnection("<Enter your valid connection string here>");  
            connection.Open();  
            NpgsqlCommand cmd = new NpgsqlCommand("SELECT * FROM tablename", connection);  
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(cmd);  
            DataTable dt = new DataTable();  
            da.Fill(dt);  
            connection.Close();  
            return dt;  
        }  
    }  
}
```

5. In the **Get()** method of the **PivotController.cs** file, the **GetPostgreSQLResult** method is used to retrieve the PostgreSQL data as a **DataTable**, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```
`c#
```

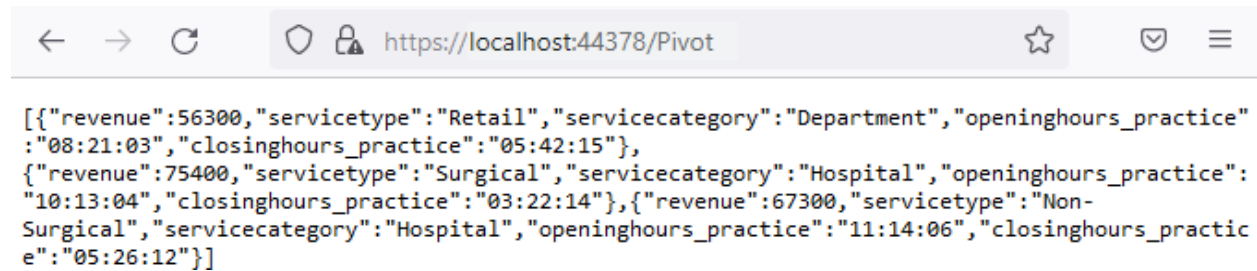
```

using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System.Data;
using Npgsql;
namespace MyWebService.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class PivotController : ControllerBase
    {
        [HttpGet(Name = "GetPostgreSQLResult")]
        public object Get()
        {
            return JsonConvert.SerializeObject(GetPostgreSQLResult());
        }
        public dynamic GetPostgreSQLResult()
        {
            // Replace with your own connection string.
            NpgsqlConnection connection = new NpgsqlConnection("<Enter your valid connection string here>");
            connection.Open();
            NpgsqlCommand cmd = new NpgsqlCommand("SELECT * FROM tablename", connection);
            NpgsqlDataAdapter da = new NpgsqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            da.Fill(dt);
            connection.Close();
            return dt;
        }
    }
}

```

6. Run the web application and it will be hosted within the URL <https://localhost:44378/>.

7. Finally, the retrieved data from PostgreSQL database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:44378/Pivot>, as shown in the browser page below.



Connecting the Pivot Table to a PostgreSQL database using the Web API service

1. Create a simple Javascript Pivot Table by following the **"Getting Started"** documentation [link](#).
2. Map the hosted Web API's URL link `https://localhost:44378/Pivot` to the Pivot Table component in `index.js` by using the `url` property under `dataSourceSettings`.

```
`javascript
import { PivotView, FieldList } from '@syncfusion/ej2-pivotview';
PivotView.Inject(FieldList);
let pivotObj: PivotView = new PivotView({
  dataSourceSettings: {
    url: 'https://localhost:44378/Pivot',
    //Other codes here...
  }
});
pivotObj.appendTo('#PivotView');
```

3. Frame and set the report based on the data retrieved from the PostgreSQL database.

```
`javascript
import { PivotView, FieldList } from '@syncfusion/ej2-pivotview';
PivotView.Inject(FieldList);
let pivotObj: PivotView = new PivotView({
  dataSourceSettings: {
    url: 'https://localhost:44378/Pivot',
    enableSorting: true,
    columns: [{ name: 'openinghourspractice' }, { name: 'closinghourspractice' }],
    values: [{ name: 'revenue' }],
    rows: [{ name: 'servicetype' }, { name: 'servicecategory' }],
  },
  showFieldList: true,
```

```
width: '100%',
height: 350,
});
pivotObj.appendTo('#PivotView');
`
```

When you run the sample, the resulting pivot table will look like this:

	▶ 08:21:03	▶ 10:13:04	▶ 11:14:06	Grand Total
▶ Non-Surgical			67300	67300
▶ Retail	56300			56300
▶ Surgical		75400		75400
Grand Total	56300	75400	67300	199000

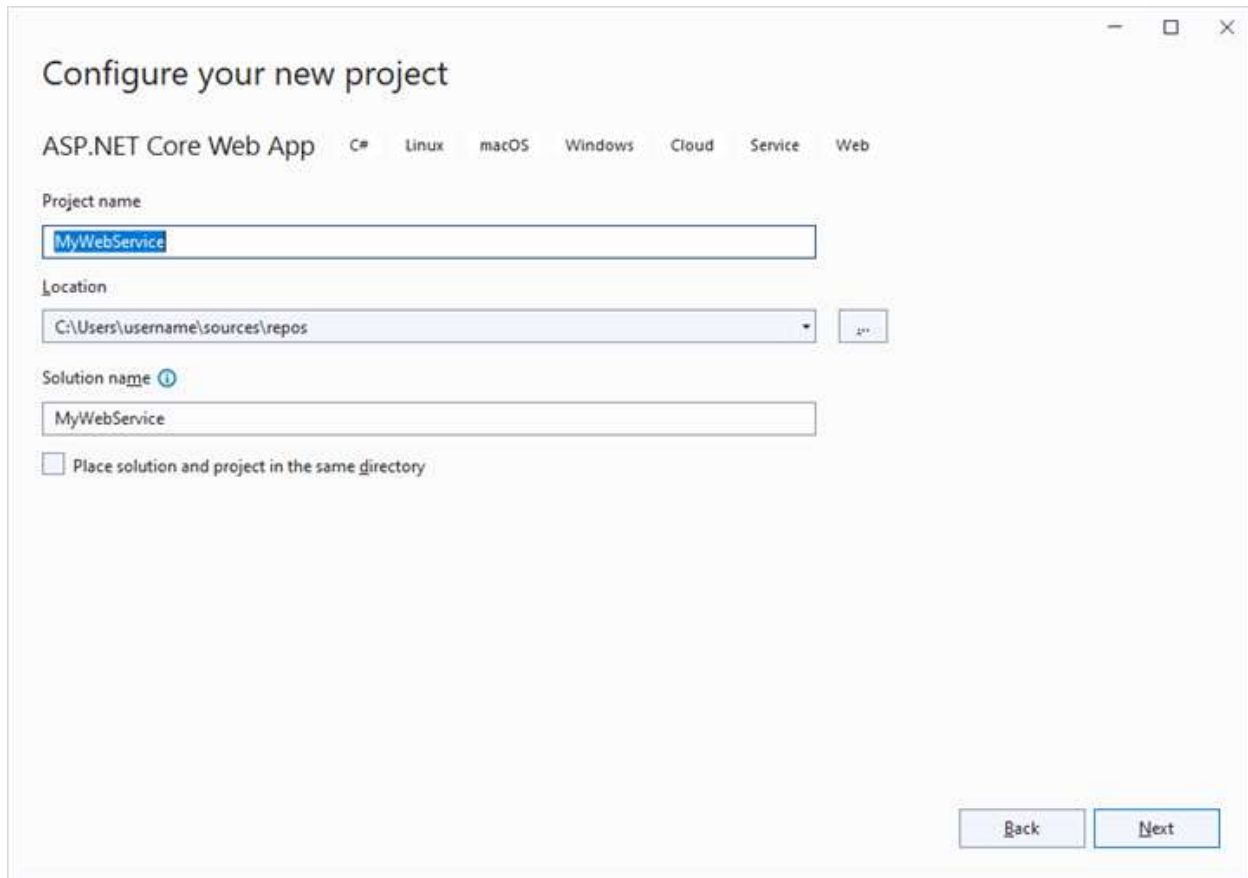
Explore our Javascript Pivot Table sample and ASP.NET Core Web Application to extract data from a PostgreSQL database and bind to the Pivot Table in [this](#) GitHub repository.

Oracle in EJ2 JavaScript Pivotview control

This section describes how to retrieve data from Oracle database using [Oracle Managed Data Access](#) library and bind it to the Pivot Table via a Web API controller.

Create a Web API service to fetch Oracle data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).



Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name
MyWebService

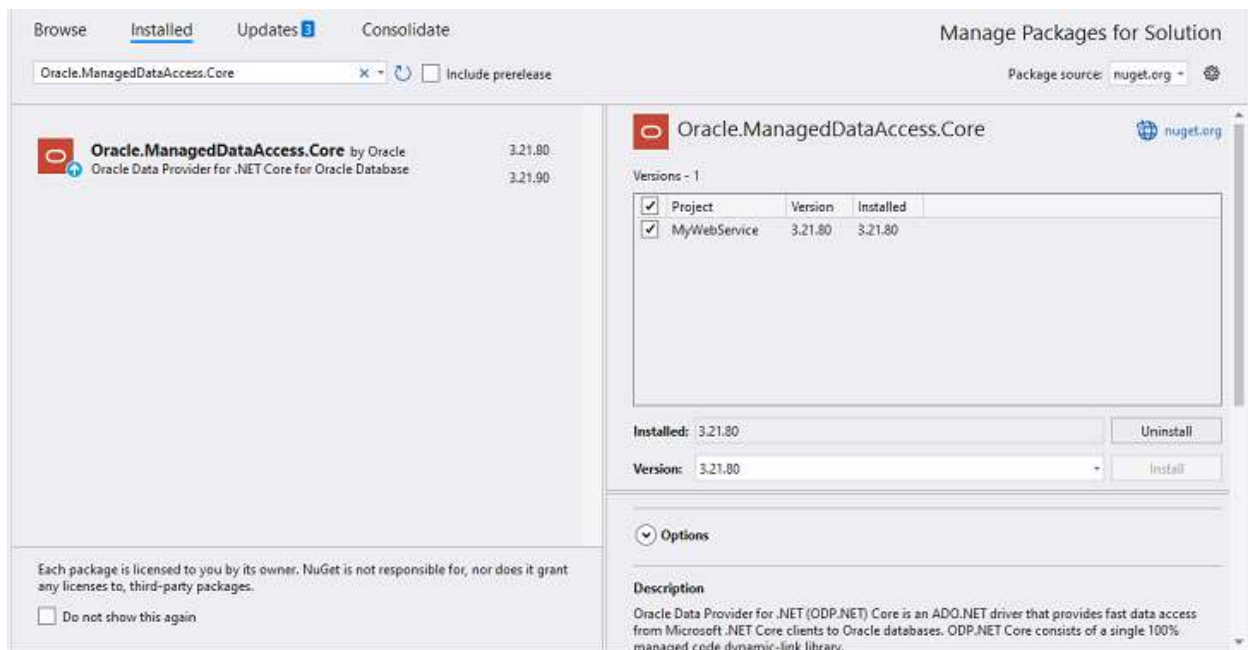
Location
C:\Users\username\source\repos

Solution name
MyWebService

☐ Place solution and project in the same directory

Back Next

2. To connect a Oracle Server using the **Oracle.ManagedDataAccess.Client** in our application, we need to install the [Oracle.ManagedDataAccess.Core](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Oracle.ManagedDataAccess.Core** and install it.



Browse Installed Updates Consolidate Manage Packages for Solution

Oracle.ManagedDataAccess.Core Include prerelease Package source: nuget.org

Oracle.ManagedDataAccess.Core by Oracle
Oracle Data Provider for .NET Core for Oracle Database

3.21.80
3.21.90

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
☐ Do not show this again

Oracle.ManagedDataAccess.Core nuget.org

Versions - 1

Project	Version	Installed
MyWebService	3.21.80	3.21.80

Installed: 3.21.80 Uninstall

Version: 3.21.80 Install

Options

Description
Oracle Data Provider for .NET (ODP.NET) Core is an ADO.NET driver that provides fast data access from Microsoft .NET Core clients to Oracle databases. ODP.NET Core consists of a single 100% managed code dynamic-link library.

3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **OracleConnection** helps to connect the Oracle database. Next, using **OracleCommand** and **OracleDataAdapter** you can process the desired Oracle query string and retrieve data from the database. The **Fill** method of the **OracleDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```
`c#  
using Microsoft.AspNetCore.Mvc;  
using Newtonsoft.Json;  
using Oracle.ManagedDataAccess.Client;  
using System.Data;  
namespace MyWebService.Controllers  
{  
    [ApiController]  
    [Route("[controller]")]  
    public class PivotController : ControllerBase  
    {  
        private static DataTable FetchOracleResult()  
        {  
            // Replace with your own connection string.  
            string connectionString = "<Enter your valid connection string here>";  
            OracleConnection oracleConnection = new OracleConnection(connectionString);  
            oracleConnection.Open();  
            OracleCommand command = new OracleCommand("SELECT * FROM EMPLOYEES", oracleConnection);  
            OracleDataAdapter dataAdapter = new OracleDataAdapter(command);  
            DataTable dataTable = new DataTable();  
            dataAdapter.Fill(dataTable);  
            oracleConnection.Close();  
            return dataTable;  
        }  
    }  
}
```

5. In the **Get()** method of the **PivotController.cs** file, the **FetchOracleResult()** method is used to retrieve the Oracle data, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```
`c#
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using Oracle.ManagedDataAccess.Client;
using System.Data;
namespace MyWebService.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class PivotController : ControllerBase
    {
        [HttpGet(Name = "GetOracleResult")]
        public object Get()
        {
            return JsonConvert.SerializeObject(FetchOracleResult());
        }
        private static DataTable FetchOracleResult()
        {
            // Replace with your own connection string.
            string connectionString = "<Enter your valid connection string here>";
            OracleConnection oracleConnection = new OracleConnection(connectionString);
            oracleConnection.Open();
            OracleCommand command = new OracleCommand("SELECT * FROM EMPLOYEES", oracleConnection);
            OracleDataAdapter dataAdapter = new OracleDataAdapter(command);
            DataTable dataTable = new DataTable();
            dataAdapter.Fill(dataTable);
            oracleConnection.Close();
            return dataTable;
        }
    }
}
```

6. Run the web application (aka, PivotController) and it will be hosted within the URL <https://localhost:44346>.

7. Finally, the retrieved data from Oracle database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:44346/Pivot>, as shown in the browser page below.



Connecting the Pivot Table to a Oracle database using the Web API service

1. Create a simple Javascript Pivot Table by following the **"Getting Started"** documentation [link](#).

2. Map the hosted Web API's URL link <https://localhost:44346/Pivot> to the Pivot Table component in `index.js` by using the `url` property under `dataSourceSettings`.

```
`javascript
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    url: 'https://localhost:44346/Pivot',
    //Other codes here...
  }
});
pivotObj.appendTo('#PivotView');
```

3. Frame and set the report based on the data retrieved from the Oracle database.

```
`javascript
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    url: 'https://localhost:44346/Pivot',
    enableSorting: true,
    expandAll: false,
```

```

dataSource: [],
columns: [
{ name: 'DEPARTMENT_ID', caption: 'Department ID' },
{ name: 'EMPLOYEE_NAME', caption: 'Employee Name' },
],
rows: [
{ name: 'JOB', caption: 'Job' },
{ name: 'SALARY', caption: 'Salary' }
],
values: [
{ name: 'EMPLOYEE_ID', caption: 'Employee ID' },
{ name: 'CC_EMPLOYEES', caption: 'Employees' },
{ name: 'CCTAXPERCENTAGE', caption: 'Percentage' },
],
filters: []
},
showFieldList: true,
width: '100%'
});
pivotObj.appendTo('#PivotView');
`

```

When you run the sample, the resulting pivot table will look like this:

	▶ 10	▶ 20	▶ 30	Grand Total
▶ ANALYST		15690		15690
▶ CLERK	7934	15245	7900	31079
▶ MANAGER	7782	7566	7698	23046
▶ PRESIDENT	7839			7839
▶ SALESMAN			30518	30518
Grand Total	23555	38501	46116	108172

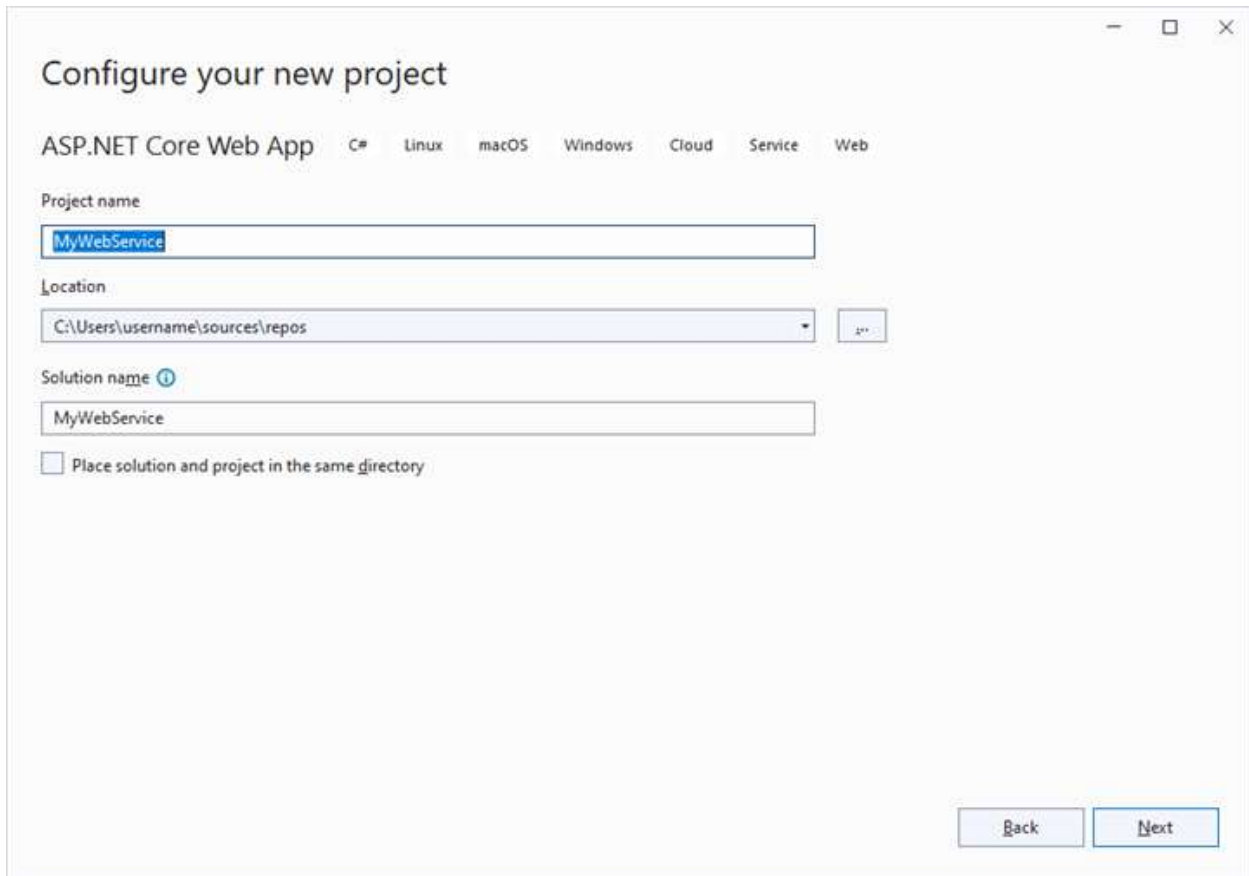
Explore our Javascript Pivot Table sample and ASP.NET Core Web Application to extract data from a Oracle database and bind to the Pivot Table in [this](#) GitHub repository.

MongoDB in EJ2 JavaScript Pivotview control

This section describes how to consume data from MongoDB database using [MongoDB Driver](#) and [MongoDB Bson](#) libraries and bind it to the Pivot Table via a Web API controller.

Create a Web API service to fetch MongoDB data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).



Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name

MyWebService

Location

C:\Users\username\source\repos

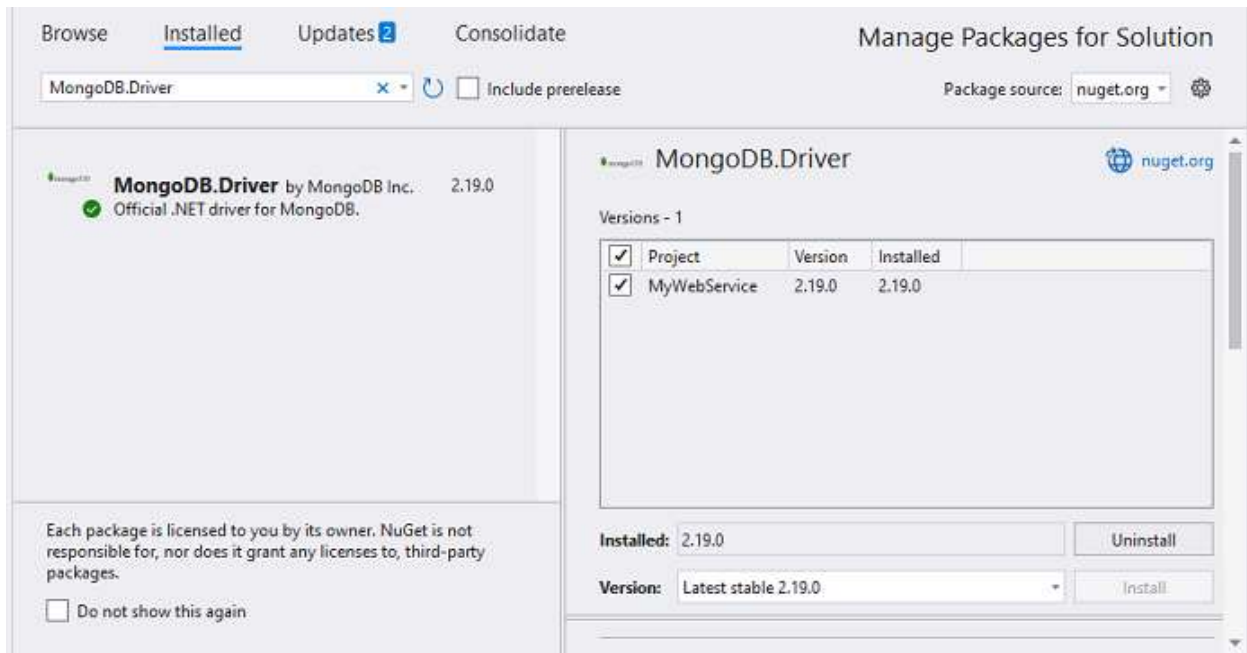
Solution name ⓘ

MyWebService

☐ Place solution and project in the same directory

Back Next

2. To connect a MongoDB Server using the **MongoDB.Driver** and **MongoDB.Bson** in our application, we need to install the [MongoDB.Driver](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **MongoDB.Driver** and install it.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **MongoClient** helps to connect the MongoDB database. Next, using the **GetDatabase** and **GetCollection** methods, you can retrieve data from the database. The **Find** method of the **IMongoDatabase** is used to populate the retrieved data into a **List**, as shown in the following code snippet.

```
`c#
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using MongoDB.Driver;
using MongoDB.Bson;
namespace MyWebService.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class PivotController : ControllerBase
    {
        private static List<ProductDetails> FetchMongoDbResult()
        {
            // Replace with your own connection string.
            string connectionString = "<Enter your valid connection string here>";
```

```

MongoClient client = new MongoClient(connectionString);
IMongoDatabase database = client.GetDatabase("sample_training");
var collection = database.GetCollection<ProductDetails>("ProductDetails");
return collection.Find(new BsonDocument()).ToList();
}

public class ProductDetails
{
    public ObjectId Id { get; set; }
    public int Sold { get; set; }
    public double Amount { get; set; }
    public string? Country { get; set; }
    public string? Products { get; set; }
    public string? Year { get; set; }
    public string? Quarter { get; set; }
}
}
}
`

```

5. In the **Get()** method of the **PivotController.cs** file, the **FetchMongoDbResult()** method is used to retrieve the MongoDB data as a **List**, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```

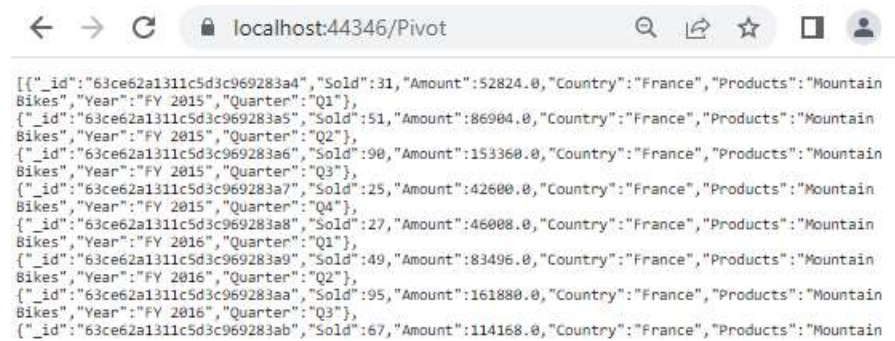
`c#
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using MongoDB.Bson;
using MongoDB.Driver;
namespace MyWebService.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class PivotController : ControllerBase
    {
        [HttpGet(Name = "GetMongoDbResult")]
        public object Get()
    }
}

```

```
{
return JsonConvert.SerializeObject(FetchMongoDbResult());
}
private static List<ProductDetails> FetchMongoDbResult()
{
// Replace with your own connection string.
string connectionString = "<Enter your valid connection string here>";
MongoClient client = new MongoClient(connectionString);
IMongoDatabase database = client.GetDatabase("sample_training");
var collection = database.GetCollection<ProductDetails>("ProductDetails");
return collection.Find(new BsonDocument()).ToList();
}
public class ProductDetails
{
public ObjectId Id { get; set; }
public int Sold { get; set; }
public double Amount { get; set; }
public string? Country { get; set; }
public string? Products { get; set; }
public string? Year { get; set; }
public string? Quarter { get; set; }
}
}
}
```

6. Run the web application and it will be hosted within the URL <https://localhost:44346>.

7. Finally, the retrieved data from MongoDB database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:44346/Pivot>, as shown in the browser page below.



Connecting the Pivot Table to a MongoDB database using the Web API service

1. Create a simple Javascript Pivot Table by following the **"Getting Started"** documentation [link](#).
2. Map the hosted Web API's URL link `https://localhost:44346/Pivot` to the Pivot Table component in `index.js` by using the `url` property under `dataSourceSettings`.

```
`javascript
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    url: 'https://localhost:44346/Pivot',
    //Other codes here...
  },
  showFieldList: true,
  width: '100%'
});
pivotObj.appendTo('#PivotView');
`
```

3. Frame and set the report based on the data retrieved from the MongoDB database.

```
`javascript
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    url: 'https://localhost:44346/Pivot',
    enableSorting: true,
    columns: [
      { name: 'Year' }
    ],
    values: [
      { name: 'Sold', caption: "Units Sold"},
      { name: 'Amount', caption: "Sold Amount"}
    ]
  }
});
```

```

],
rows: [
{ name: 'Country' },
{ name: 'Products' }
],
showFieldList: true,
width: '100%'
});
pivotObj.appendTo('#PivotView');

```

When you run the sample, the resulting pivot table will look like this:

	FY 2015		FY 2016		F
	Units Sold	Sold Amount	Units Sold	Sold Amount	U
> France	729	1160099.5	609	983317	
> Germany	528	845472	667	1067220	
> United Kingdom	782	1263109.5	640	1031630.5	
> United States	682	1085398.5	480	770362	
Grand Total	2721	4354079.5	2396	3852529.5	

Explore our Javascript Pivot Table sample and ASP.NET Core Web Application to extract data from a MongoDB database and bind to the Pivot Table in [this](#) GitHub repository.

Elasticsearch in EJ2 JavaScript Pivotview control

This section describes how to retrieve data from Elasticsearch database using [Nest](#) and bind it to the Pivot Table via a Web API controller.

Create a Web API service to fetch Elasticsearch data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).

Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name
MyWebService

Location
C:\Users\username\source\repos

Solution name
MyWebService

☐ Place solution and project in the same directory

Back Next

2. To connect a Elasticsearch Server using the **NEST** in our application, we need to install the [NEST](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **NEST** and install it.

Browse Installed Updates Consolidate Manage Packages for Solution

NEST x Include prerelease Package source: nuget.org

NEST by Elastic and contributors 7.17.5
Strongly typed interface to Elasticsearch. Fluent and classic object initializer mappings of requests and

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
☐ Do not show this again

NEST

Versions - 1

Project	Version	Is Installed
MyWebService	7.17.5	7

Installed: 7.17.5 Uninstall

Version: Latest stable 7.17.5 Install

3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **ElasticClient** helps to connect the Elasticsearch database. Next, using **Search** method you can query your Elasticsearch index and retrieve results from the database.

5. In the **Get()** method of the **PivotController.cs** file, the **FetchElasticsearchData** method is used to retrieve the Elasticsearch data, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```
`c#  
using Microsoft.AspNetCore.Mvc;  
using Nest;  
using Newtonsoft.Json;  
namespace MyWebService.Controllers  
{  
    [ApiController]  
    [Route("[controller]")]  
    public class PivotController : ControllerBase  
    {  
        [HttpGet(Name = "GetElasticSearchData")]  
        public object Get()  
        {  
            return JsonConvert.SerializeObject(FetchElasticsearchData());  
        }  
        private static object FetchElasticsearchData()  
        {  
            // Replace with your own connection string.  
            var connectionString = "<Enter your valid connection string here>";  
            var uri = new Uri(connectionString);  
            var connectionSettings = new ConnectionSettings(uri);  
            var client = new ElasticClient(connectionSettings);  
            var searchResponse = client.Search<object>(s => s  
                .Index("product")  
                .Size(1000)  
            );  
            return searchResponse.Documents;  
        }  
    }  
}
```

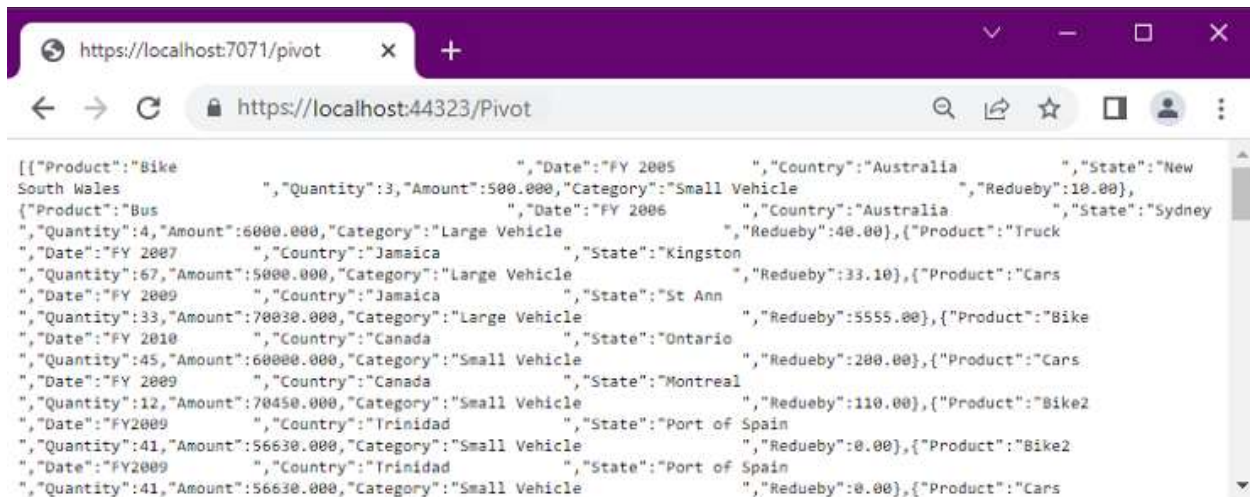
```

}
}
}
,

```

6. Run the web application and it will be hosted within the URL <https://localhost:44323>.

7. Finally, the retrieved data from Elasticsearch database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:44323/Pivot>, as shown in the browser page below.



Connecting the Pivot Table to an Elasticsearch database using the Web API service

1. Create a simple Javascript Pivot Table by following the **"Getting Started"** documentation [link](#).

2. Map the hosted Web API's URL link <https://localhost:44323/Pivot> to the Pivot Table in **index.js** by using the **url** property under **dataSourceSettings**.

```

`javascript
var pivotObj = new ej.pivotview.PivotView({
dataSourceSettings: {
url: 'https://localhost:44323/Pivot',
//Other codes here...
}
});
pivotObj.appendTo('#PivotView');
,

```

3. Frame and set the report based on the data retrieved from the Elasticsearch database.

```

`javascript
var pivotObj = new ej.pivotview.PivotView({

```



```

dataSourceSettings: {
  url: 'https://localhost:44323/Pivot',
  expandAll: false,
  enableSorting: true,
  columns: [{ name: 'Product' }],
  values: [{ name: 'Quantity' }, { name: 'Amount', caption: 'Sold Amount' }],
  rows: [{ name: 'Country' }, { name: 'State' }],
  formatSettings: [{ name: 'Amount', format: 'C0' }],
  filters: []
},
showFieldList: true,
width: '100%'
});
pivotObj.appendTo('#PivotView');

```

When you run the sample, the resulting pivot table will look like this:

	▶ Large Vehicle		▶ Small Vehicle		Grand Total	
	Quantity	Amount	Quantity	Amount	Quantity	Am
▶ Australia	16	24000	12	2000	28	
▶ Canada			420	1649000	420	
▶ Jamaica	400	300120			400	
▶ Trinidad			328	453040	328	
Grand Total	416	324120	760	2104040	1176	

Explore our Javascript Pivot Table sample and ASP.NET Core Web Application to extract data from a Elasticsearch database and bind to the Pivot Table in [this](#) GitHub repository.

Snowflake in EJ2 JavaScript Pivotview control

This section describes how to retrieve data from Snowflake database using [Snowflake Data](#) library and bind it to the Pivot Table via a Web API controller.

Create a Web API service to fetch Snowflake data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).

Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name
MyWebService

Location
C:\Users\username\source\repos

Solution name
MyWebService

☐ Place solution and project in the same directory

Back Next

2. To connect a Snowflake Server using the **Snowflake.Data.Client** in our application, we need to install the [Snowflake.Data](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Snowflake.Data** and install it.

Manage Packages for Solution

Browse Installed Updates Consolidate

Snowflake.Data x Include prerelease Package source: nuget.org

Snowflake.Data by howryu, tchen 2.0.22
Snowflake Connector for .NET

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
☐ Do not show this again

Snowflake.Data nuget.org

Versions - 1

Project	Version	Installed
MyWebService	2.0.22	2.0.22

Installed: 2.0.22 Uninstall

Version: Latest stable 2.0.22 Install

Options

Description
Snowflake Connector for .NET

3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **SnowflakeDbConnection** helps to connect the Snowflake database. Next, using **SnowflakeDbDataAdapter** you can process the desired Snowflake query string and retrieve data from the database. The **Fill** method of the **SnowflakeDbDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```
`c#
using Microsoft.AspNetCore.Mvc;
using Snowflake.Data.Client;
using Newtonsoft.Json;
using System.Data;
namespace MyWebService.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class PivotController : ControllerBase
    {
        public static DataTable FetchSnowflakeResult()
        {
            using (SnowflakeDbConnection snowflakeConnection = new SnowflakeDbConnection())
            {
                // Replace with your own connection string.
                snowflakeConnection.ConnectionString = "<Enter your valid connection string here>";
                snowflakeConnection.Open();
                SnowflakeDbDataAdapter adapter = new SnowflakeDbDataAdapter("select * from CALL_CENTER",
                    snowflakeConnection);
                DataTable dataTable = new DataTable();
                adapter.Fill(dataTable);
                snowflakeConnection.Close();
                return dataTable;
            }
        }
    }
}
```

5. In the **Get()** method of the **PivotController.cs** file, the **FetchSnowflakeResult** method is used to retrieve the Snowflake data as a **DataTable**, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```
`c#  
using Microsoft.AspNetCore.Mvc;  
using Snowflake.Data.Client;  
using Newtonsoft.Json;  
using System.Data;  
namespace MyWebService.Controllers  
{  
    [ApiController]  
    [Route("[controller]")]  
    public class PivotController : ControllerBase  
    {  
        [HttpGet(Name = "GetSnowflakeResult")]  
        public object Get()  
        {  
            return JsonConvert.SerializeObject(FetchSnowflakeResult());  
        }  
        public static DataTable FetchSnowflakeResult()  
        {  
            using (SnowflakeDbConnection snowflakeConnection = new SnowflakeDbConnection())  
            {  
                // Replace with your own connection string.  
                snowflakeConnection.ConnectionString = "<Enter your valid connection string here>";  
                snowflakeConnection.Open();  
                SnowflakeDbDataAdapter adapter = new SnowflakeDbDataAdapter("select * from CALL_CENTER",  
                    snowflakeConnection);  
                DataTable dataTable = new DataTable();  
                adapter.Fill(dataTable);  
                snowflakeConnection.Close();  
                return dataTable;  
            }  
        }  
    }  
}
```

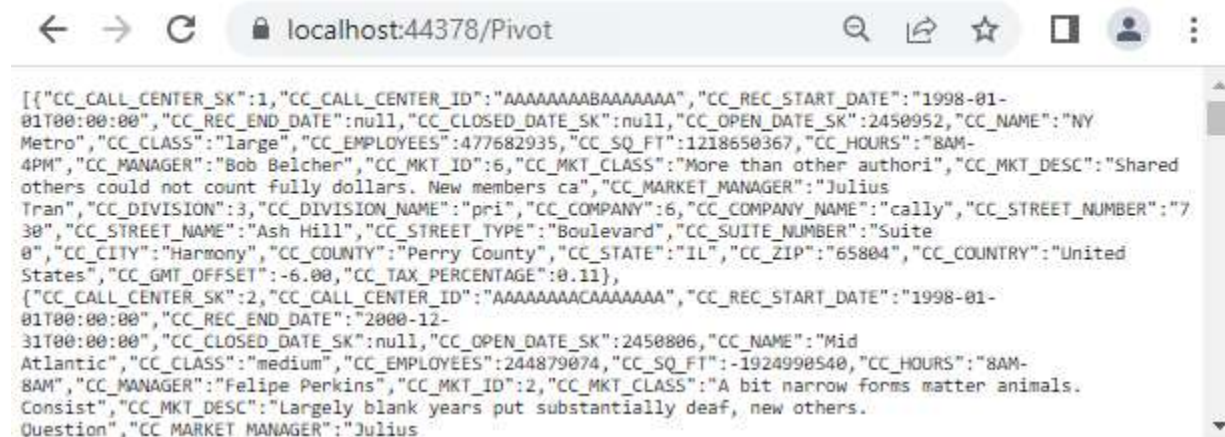
```

}
}
,

```

6. Run the web application and it will be hosted within the URL <https://localhost:44378/>.

7. Finally, the retrieved data from Snowflake database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:44378/Pivot>, as shown in the browser page below.



Connecting the Pivot Table to a Snowflake database using the Web API service

1. Create a simple Javascript Pivot Table by following the “Getting Started” documentation [link](#).
2. Map the hosted Web API's URL link <https://localhost:44378/Pivot> to the Pivot Table component in `index.js` by using the `url` property under `dataSourceSettings`.

```

`javascript
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    url: 'https://localhost:44378/Pivot',
    //Other codes here...
  }
});
pivotObj.appendTo('#PivotView');
,

```

3. Frame and set the report based on the data retrieved from the Snowflake database.

```

`javascript
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    url: 'https://localhost:44378/Pivot',

```

```

enableSorting: true,
expandAll: false,
dataSource: [],
columns: [
{ name: 'CC_COUNTRY', caption: 'Country' }
],
rows: [
{ name: 'CC_STATE', caption: 'State' },
{ name: 'CC_CITY', caption: 'City' }
],
values: [
{ name: 'CC_COMPANY', caption: 'Company' },
{ name: 'CC_EMPLOYEES', caption: 'Employees' },
{ name: 'CCTAXPERCENTAGE', caption: 'Tax percentage' },
],
filters: []
},
showFieldList: true,
width: '100%'
});
pivotObj.appendTo('#PivotView');
`

```

When you run the sample, the resulting pivot table will look like this:

	United States			Grand Total		
	Company	Employees	Percentage	Company	Employees	Percentage
▶ CA	16	413279139	0.4800000000...	16	413279139	0.480000...
▶ CO	5	1160903623	0.18	5	1160903623	
▶ FL	15	1162362540	0.33	15	1162362540	
▶ GA	20	2358562323	0.12	20	2358562323	
▶ IL	8	707852970	0.1699999999...	8	707852970	0.169999...
▶ KS	9	1842349179	0.19	9	1842349179	
▶ LA	6	528468075	0.16	6	528468075	

Explore our Javascript Pivot Table sample and ASP.NET Core Web Application to extract data from a Snowflake database and bind to the Pivot Table in [this](#) GitHub repository.

Olap in EJ2 JavaScript Pivotview control

Getting Started

This section explain steps to create a simple Essential JS 2 **Pivot Table** with OLAP data source in a JavaScript application.

Dependencies

The following list of dependencies are required to use the pivot table component in your application.

```
`javascript
|-- @syncfusion/ej2-pivotview
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-excel-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-calendars
|-- @syncfusion/ej2-charts
|-- @syncfusion/ej2-svg-base
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-grids
`
```

Setup for local environment

Refer the following steps for setup your local environment.

Step 1: Create a root folder **myapp** for your application.

Step 2: Create **myapp/resources** folder to store local scripts and styles files.

Step 3: Create **myapp/index.js** and **myapp/index.html** files for initializing Essential JS 2 pivot table control.

Adding Syncfusion resources

The Essential JS 2 pivot table control can be initialized by using either of the following ways.

- Using local script and style.
- Using CDN link for script and style.

Using local script and style

You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

After installing the Essential JS 2 product build, you can copy the pivot table and its [dependency](#) scripts and styles file into the **resources/scripts** and **resources/styles** folder respectively.

Refer the below location to find pivot table's script and style file.

Syntax:

Script: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js**

Styles: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css**

Example:

Script: C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-pivotview/dist/global/ej2-pivotview.min.js

Styles: C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-pivotview/styles/material.css

After copying the files, then you can refer the pivot table and its [dependency](#) scripts and styles into the **index.html** file.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 pivot table's dependent material theme -->
<link href="resources/ej2-base/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-buttons/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-calendars/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-dropdowns/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-inputs/styles/material.css" rel="stylesheet" type="text/css"/>
```



```

<link href="resources/ej2-lists/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-popups/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-navigations/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/ej2-grids/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 pivot table's material theme -->
<link href="resources/ej2-pivotview/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 pivot table's dependent scripts -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-buttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-file-utils.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-compression.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-excel-export.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-lists.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-pdf-export.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-svg-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-inputs.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-calendars.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-dropdowns.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-navigations.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-charts.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-grids.min.js" type="text/javascript"></script>
<!-- Essential JS 2 pivot table's global script -->
<script src="resources/scripts/ej2-pivotview.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>

```

In this illustration, we have referred **material** theme. You can also refer other themes like bootstrap, fabric, high-contrast etc.,

Using CDN link for script and style

Using CDN link, you can directly refer the pivot table and its [dependency](#) scripts and styles into the **index.html**.

Refer the pivot table's CDN links as below

Syntax:

Script: `http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Styles: `http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

Example:

Script: <http://cdn.syncfusion.com/ej2/ej2-pivotview/dist/global/ej2-pivotview.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-pivotview/styles/material.css>

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 pivot table's dependent material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-calendars/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 pivot table's material theme -->
```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-pivotview/styles/material.css" rel="stylesheet"
type="text/css"/>

<!-- Essential JS 2 pivot table's dependent scripts -->

<script src="http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-buttons/dist/global/ej2-buttons.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-file-utils/dist/global/ej2-file-utils.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-compression/dist/global/ej2-compression.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-excel-export/dist/global/ej2-excel-export.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-lists/dist/global/ej2-lists.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-pdf-export/dist/global/ej2-pdf-export.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-popups.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-svg-base/dist/global/ej2-svg-base.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-inputs/dist/global/ej2-inputs.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-calendars/dist/global/ej2-calendars.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-dropdowns/dist/global/ej2-dropdowns.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-charts/dist/global/ej2-charts.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js"
type="text/javascript"></script>

<!-- Essential JS 2 pivot table's global script -->

<script src="http://cdn.syncfusion.com/ej2/ej2-pivotview/dist/global/ej2-pivotview.min.js"
type="text/javascript"></script>

</head>
```

```
<body>
</body>
</html>
`
```

In this illustration, we have referred **material** theme. You can also refer other themes like bootstrap, fabric, high-contrast etc.,

Adding pivot table control

Next you can start adding pivot table control to the application. To get started, add **div** element for pivot table control in **index.html**. Then refer the **index.js** file into the **index.html** file.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 pivot table's dependent material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-calendars/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 pivot table's material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-pivotview/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 pivot table's dependent scripts -->
```

```
<script src="http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-buttons/dist/global/ej2-buttons.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-file-utils/dist/global/ej2-file-utils.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-compression/dist/global/ej2-compression.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-excel-export/dist/global/ej2-excel-export.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-lists/dist/global/ej2-lists.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-pdf-export/dist/global/ej2-pdf-export.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-popups.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-svg-base/dist/global/ej2-svg-base.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-inputs/dist/global/ej2-inputs.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-calendars/dist/global/ej2-calendars.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-dropdowns/dist/global/ej2-dropdowns.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-charts/dist/global/ej2-charts.min.js"
type="text/javascript"></script>

<script src="http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js"
type="text/javascript"></script>

<!-- Essential JS 2 pivot table's global script -->

<script src="http://cdn.syncfusion.com/ej2/ej2-pivotview/dist/global/ej2-pivotview.min.js"
type="text/javascript"></script>

</head>

<body>

<!-- Add the HTML <div> element for pivot table -->
```

```
<div id="PivotTable"></div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Next we need to initialize pivot table component using **`ej.pivotview.PivotView()`** instance as follows.

```
`javascript
var pivotTableObj = new ej.pivotview.PivotView();
```

After initialization, add the the following code in **`index.js`** file to populate pivot table with a sample OLAP data source. Refer [here](#) to know the more details about OLAP data binding.

```
`javascript
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    catalog: 'Adventure Works DW 2008 SE',
    cube: 'Adventure Works',
    providerType: 'SSAS',
    enableSorting: true,
    url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
    localeIdentifier: 1033
  }
});
pivotTableObj.appendTo('#PivotTable');
```

Adding OLAP cube elements to row, column, value and filter axes

Now that pivot table is initialized and assigned with sample OLAP data source, will further move to showcase the component by organizing appropriate [OLAP cube elements](#) in [rows](#), [columns](#), [values](#) and [filters](#) axes.

In [dataSourceSettings](#) property, four major axes [rows](#), [columns](#), [values](#) and [filters](#) plays a vital role in defining and organizing [OLAP cube elements](#) from the bound data source, to render the entire pivot table component in a desired format.

[rows](#) – Collection of [OLAP cube elements](#) (such as Hierarchies, NamedSet, Calculated Members etc.,) that needs to be displayed in row axis of the pivot table.

[columns](#) – Collection of [OLAP cube elements](#) (such as Hierarchies, NamedSet, Calculated Members etc.,) that needs to be displayed in column axis of the pivot table.

values – Collection of [OLAP cube elements](#) (such as Measures, Calculated Measures) that needs to be displayed as aggregated numeric values in the pivot table.

filters - Collection of [OLAP cube elements](#) (such as Hierarchies and Calculated Members) that would act as master filter over the data bound in row, column and value axes of the pivot table.

In-order to define each [OLAP cube element](#) in the respective axis, the following basic properties should be set.

- **name**: It allows to set the unique name of the hierarchies, named set, measures, calculated members etc., from the bound OLAP data source. It's casing should match exactly like in the data source and if not set properly, the pivot table will be rendered as empty.
- **caption**: It allows to set the caption, which is the alias name of the unique name that needs to be displayed in the pivot table. If not provided, unique name will be displayed.

In this sample, "Product Categories" is added in column, "Customer Geography" in row, and "Customer Count" and "Internet Sales Amount" in value axes respectively.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    catalog: 'Adventure Works DW 2008 SE',
    cube: 'Adventure Works',
    providerType: 'SSAS',
    enableSorting: true,
    url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
    localeIdentifier: 1033,
    rows: [
      { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
    ],
    columns: [
      { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
      { name: '[Measures]', caption: 'Measures' },
    ],
    values: [
      { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
      { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
    ]
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Applying formatting to measures

Formatting defines a way in which values should be displayed in pivot table. For example, format “C0” denotes the values should be displayed in currency pattern without decimal points. To do so, define the [formatSettings](#) with its [name](#) and [format](#) properties. In this sample, the [name](#) property is set as

"[Measures].[Internet Sales Amount]", a measure from value axis and its format is set as "C0". Likewise, we can set format for other measures as well.

Only measures from value axis, which is in the form of numeric data values are applicable for formatting.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    catalog: 'Adventure Works DW 2008 SE',
    cube: 'Adventure Works',
    providerType: 'SSAS',
    enableSorting: true,
    url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
    localeIdentifier: 1033,
    rows: [
      { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
    ],
    columns: [
      { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
      { name: '[Measures]', caption: 'Measures' },
    ],
    values: [
      { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
      { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
    ],
    formatSettings: [{ name: '[Measures].[Internet Sales Amount]',
format: 'C0' }]
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable grouping bar

The Grouping Bar feature automatically populates [OLAP cube elements](#) from the bound data source and allows end users to drag [OLAP cube elements](#) between different axes such as [rows](#), [columns](#), [values](#) and [filters](#), and change pivot view at runtime. Sorting, filtering and removing of elements is also possible. It can be enabled by setting the [showGroupingBar](#) property to **true** as follows.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        catalog: 'Adventure Works DW 2008 SE',
        cube: 'Adventure Works',
        providerType: 'SSAS',
        enableSorting: true,
        url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
        localeIdentifier: 1033,
    }
});

```

```

        rows: [
            { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
        ],
        columns: [
            { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
            { name: '[Measures]', caption: 'Measures' },
        ],
        values: [
            { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
            { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
        ],
        filters: [
            { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
        ],
        filterSettings: [
            {
                name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                levelCount: 3
            }
        ]
    },
    showGroupingBar: true,
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable pivot field list

The component provides a built-in Field List similar to Microsoft Excel. It allows you to add or remove [OLAP cube elements](#) and also rearrange the [OLAP cube elements](#) between different axes, including [rows](#), [columns](#), [values](#) and [filters](#) along with filter and sort options dynamically at runtime. It can be enabled by setting the [showFieldList](#) property to **true** as follows.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        catalog: 'Adventure Works DW 2008 SE',
        cube: 'Adventure Works',
        providerType: 'SSAS',
        enableSorting: true,
        url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
        localeIdentifier: 1033,
        rows: [
            { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
        ],
        columns: [

```

```

        { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
        { name: '[Measures]', caption: 'Measures' },
    ],
    values: [
        { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
        { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
    ],
    filters: [
        { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
    ],
    filterSettings: [
        {
            name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
'[Date].[Fiscal].[Fiscal Year].&[2005]'],
            levelCount: 3
        }
    ]
},
showFieldList: true,
height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exploring filter axis

The filter axis contains collection of [OLAP cube elements](#) such as hierarchies and calculated members that would act as master filter over the data bound in [rows](#), [columns](#) and [values](#) axes of the pivot table. The [OLAP cube elements](#) along with filter members could be set to filter axis either through report via code behind or by dragging and dropping [OLAP cube elements](#) from other axes to filter axis via grouping bar or field list at runtime.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        catalog: 'Adventure Works DW 2008 SE',
        cube: 'Adventure Works',
        providerType: 'SSAS',
        enableSorting: true,
        url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
        localeIdentifier: 1033,
        rows: [
            { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
        ],
        columns: [
            { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
            { name: '[Measures]', caption: 'Measures' },
        ],
        values: [

```

```

        { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
        { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
    ],
    filters: [
        { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
    ],
    filterSettings: [
        {
            name: '[Date].[Fiscal]', items:
[ '[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
            '[Date].[Fiscal].[Fiscal Year].&[2005]' ],
            levelCount: 3
        }
    ]
},
showGroupingBar: true,
showFieldList: true,
height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

```

```

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Calculated field

The calculated field allows user to insert or add a new calculated field based on the available [OLAP cube elements](#) from the bound data source. Calculated fields are nothing but customized dimensions or measures that are newly created based on the user-defined expression.

The two types of calculated fields are as follows:

- **Calculated Measure** – Creates a new measure through user-defined expression.
- **Calculated Dimension** – Creates a new dimension through user-defined expression.

It can be customized using the [calculatedFieldsSettings](#) property through code behind. The setting required for calculate field feature at code behind are:

- [name](#): It allows to set the unique name for new calculated field.
- [formula](#): It allows to set the user-defined expression.
- [hierarchyUniqueName](#): It allows to specify dimension unique name whose hierarchies alone should be used in the expression. This will be applicable only for calculated dimension.
- [formatString](#): It allows to set the format string for the resultant calculated field.

You need to set [isCalculatedField](#) property to true, while adding calculated fields to respective axis through code behind.

Also calculated fields can be added at run time through the built-in dialog. The dialog can be enabled by setting the [allowCalculatedField](#) property to **true** as follows. You will see a button enabled in the Field List UI automatically to invoke the calculated field dialog and perform necessary operation.

Calculated measure can be added only in value axis.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    catalog: 'Adventure Works DW 2008 SE',
    cube: 'Adventure Works',
    providerType: 'SSAS',
    enableSorting: true,
    url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
    localeIdentifier: 1033,
    rows: [
      { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
    ],
    columns: [
      { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
      { name: '[Measures]', caption: 'Measures' },
    ],
    values: [
      { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
      { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' },
      { name: 'Order on Discount', isCalculatedField: true }
    ],
    filters: [
      { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
    ],
    calculatedFieldSettings: [
      {
        name: 'BikeAndComponents',
        formula: '([Product].[Product
Categories].[Category].[Bikes] + [Product].[Product
Categories].[Category].[Components] )',
        hierarchyUniqueName: '[Product].[Product Categories]',
        formatString: 'Standard'
      },
      {
        name: 'Order on Discount',
        formula: '[Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)',
        formatString: 'Currency'
      }
    ],
    filterSettings: [
      {
        name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
'[Date].[Fiscal].[Fiscal Year].&[2005]'],
        levelCount: 3
      }
    ]
  },
  showFieldList: true,
  showGroupingBar: true,
  allowCalculatedField: true,

```

```

        height: 350
    });
    pivotTableObj.appendTo('#PivotTable');
    var btn = new ej.buttons.Button({ isPrimary: true }, '#CalculatedField');
    document.getElementById('CalculatedField').addEventListener('click', () => {
        pivotTableObj.calculatedFieldModule.createCalculatedFieldDialog();
    });

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="CalculatedField">Calculated Field</div>
        <div id="PivotTable"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }

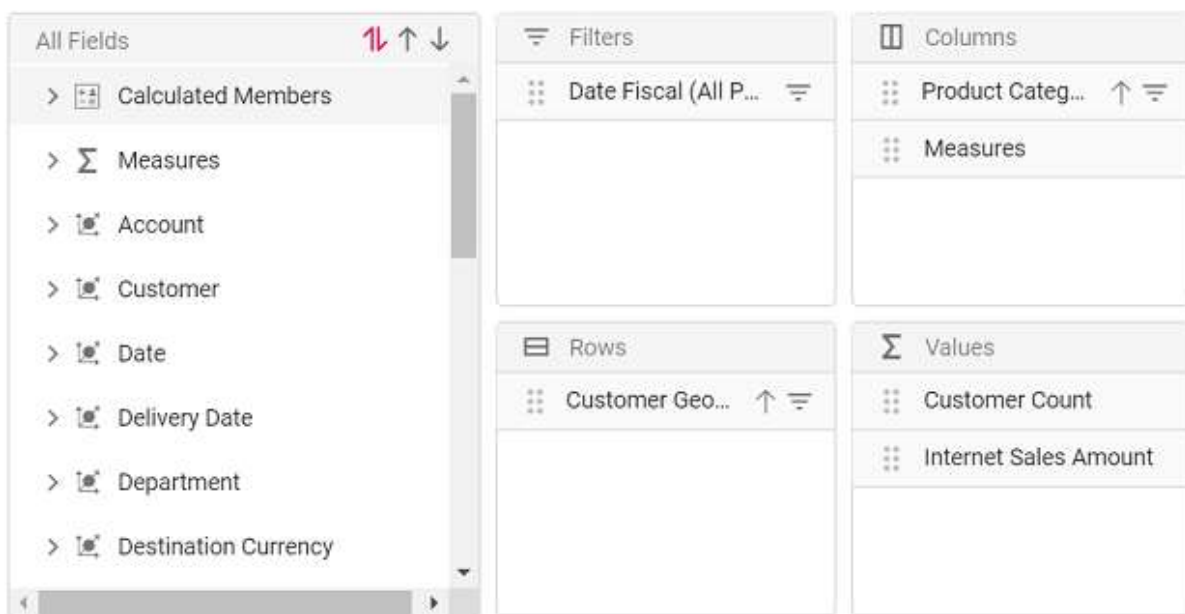
```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Users can add a calculated field at runtime through the built-in dialog by using the following steps.

Step 1: Click the "CALCULATED FIELD" button in the field list dialog positioned at the top right corner. The calculated field dialog will be opened now. Enter the name of the calculated field to be created.

Field List



CLOSE

Create Calculated Field

All Fields

- > Calculated Members
- > Measures
- > Account
- > Customer
- > Date
- > Delivery Date
- > Department
- > Destination Currency
- > Employee
- > Geography
- > Internet Sales Order Details
- > Organization
- > Product

Field Name
BikeAndComponents

Expression
Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)

Field Type
Measure

Parent Hierarchy
Account Number

Format String
Standard

Enter custom format string

CLEAR OK CANCEL

Step 2: Frame the expression by dragging and dropping the fields from the tree view on the left side of the dialog using simple arithmetic operators. **Example:** "IIF([Measures].[Internet Sales Amount]^0.5 > 100, [Measures].[Internet Sales Amount]*100, [Measures].[Internet Sales Amount]/100)". Please refer here to learn more about the supported [operators](#) and [functions](#) to frame the expression.

Create Calculated Field

All Fields

- > History
- > Large Photo
- > Model Name
- > Product
- > Product Categories
- > Product Line
- > Product Model Lines
- > Style
- > Subcategory
- > Promotion
- > Reseller
- > Reseller Sales Order Details

Field Name
BikeAndComponents

Expression
Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)
Product Categories

Field Type
Measure

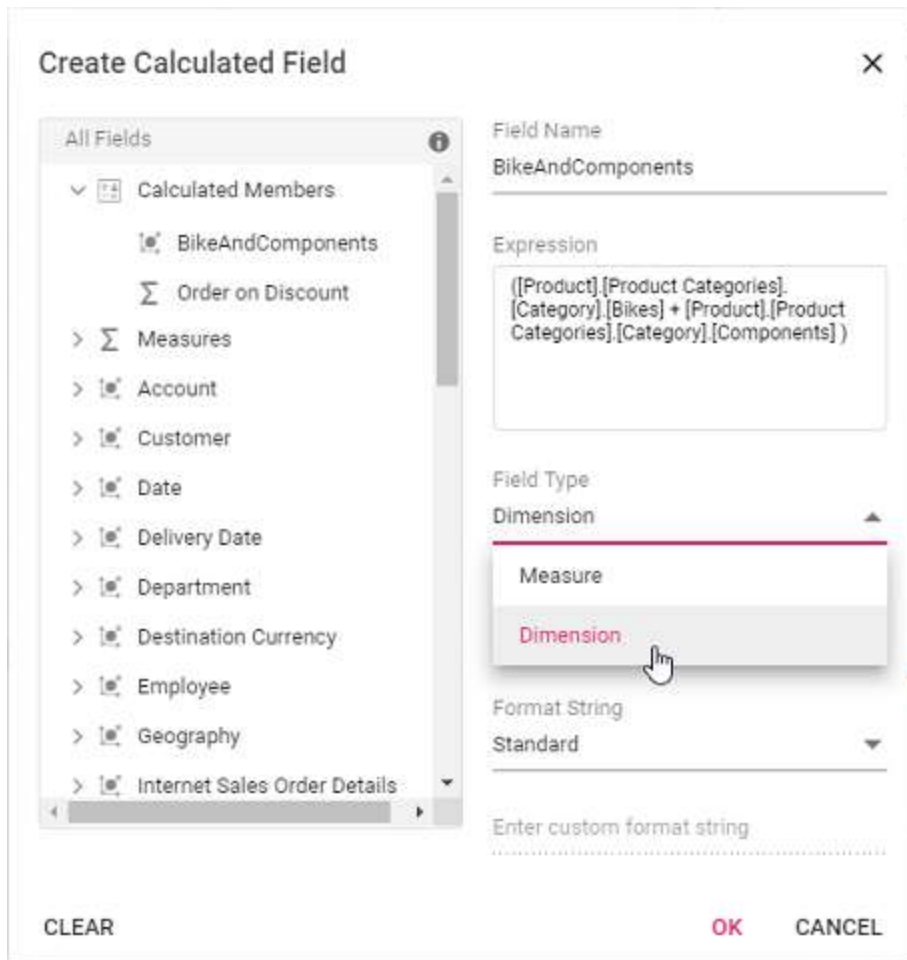
Parent Hierarchy
Account Number

Format String
Standard

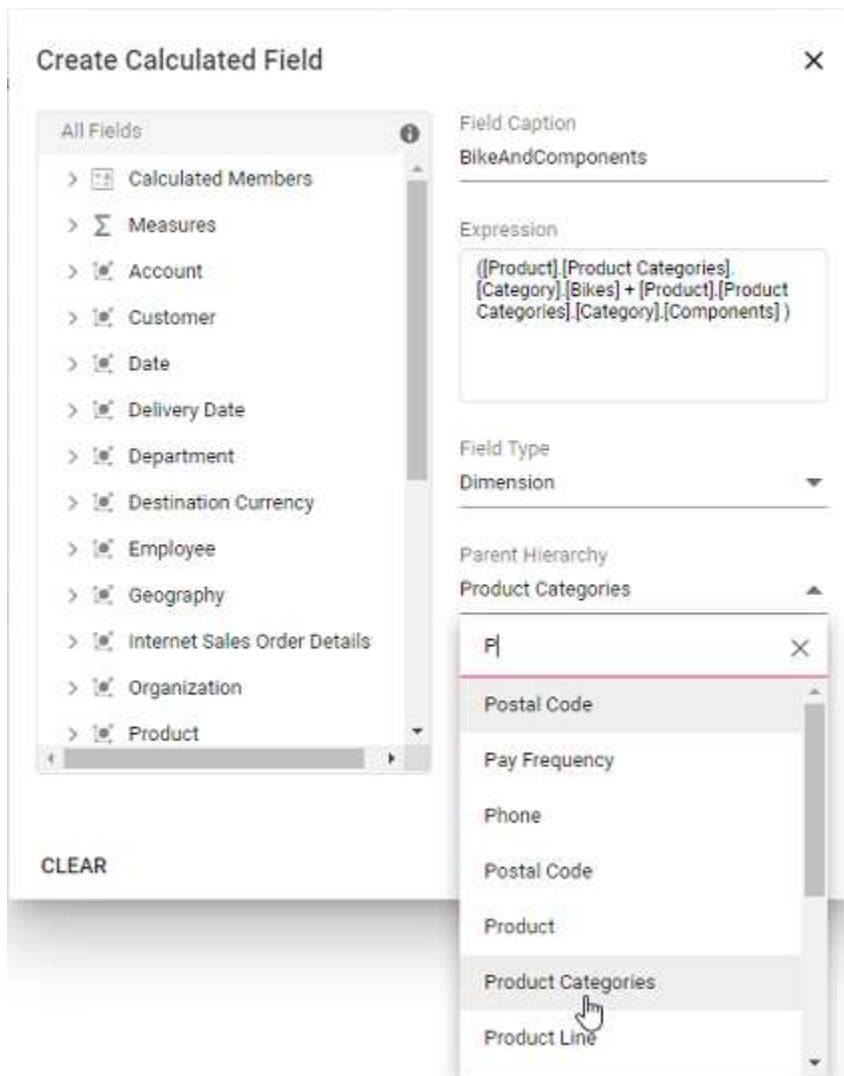
Enter custom format string

CLEAR OK CANCEL

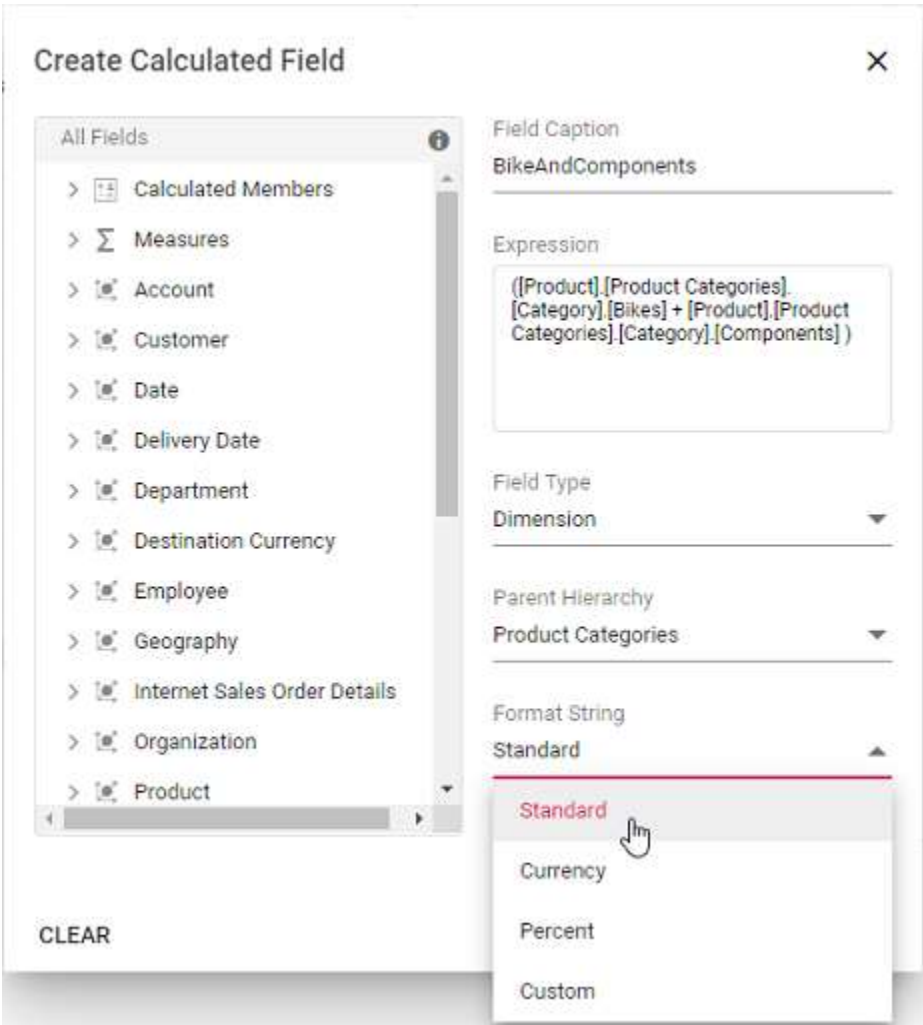
Step 3: Confirm the type of the field to be created - calculated measure or calculated dimension.



Step 4: Choose the parent hierarchy of the calculated field. NOTE: It is only applicable to the calculated dimension.



Step 5: Then select the format string from the drop-down list and finally click "OK".



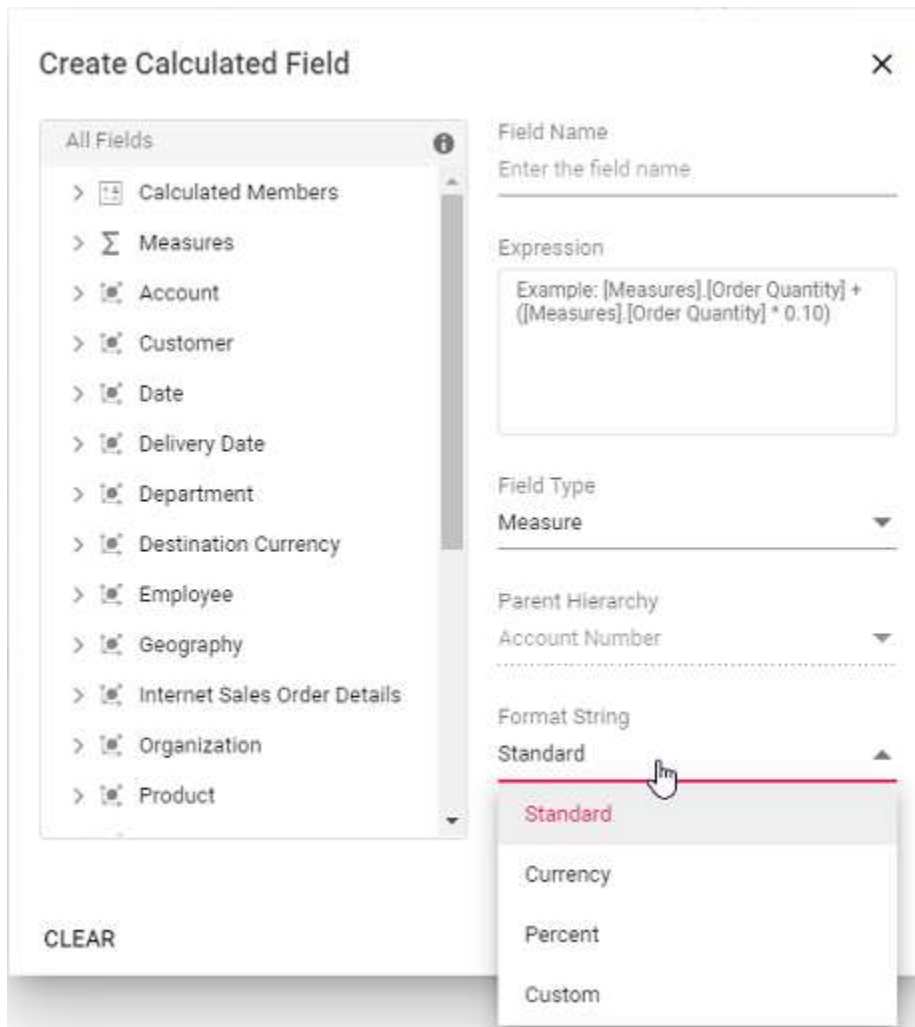
	Accessories		
	Customer Count	Internet Sales Amount	Order on Discount
▶ Australia	2,905	\$138,690.63	\$68,124.10
▶ Canada	1,230	\$103,377.85	\$68,124.10
▶ France	1,505	\$63,406.78	\$68,124.10
▶ Germany	1,535	\$62,232.59	\$68,124.10

Format String

Allows you to specify the required format string while creating new calculated field. Supported format strings are:

- **Standard** - Denotes the numeric type.
- **Currency** - Denotes the currency type.
- **Percent** - Denotes the percentage type.
- **Custom** - Denotes the custom format. For example: "###0.##0#". This shows the value "9584.3" as "9584.300."

By default, **Standard** will be selected from the drop down list.



Renaming the existing calculated field

Existing calculated field can be renamed only through the UI at runtime. To do so, open the calculated field dialog, click the target field. User can now see the existing name getting displayed in the text box at the top of the dialog. Now, change the name based on user requirement and click "OK".

<!-- markdownlint-disable MD012 -->

Create Calculated Field

All Fields

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)

Field Type

Measure

Parent Hierarchy

Account Number

Format String

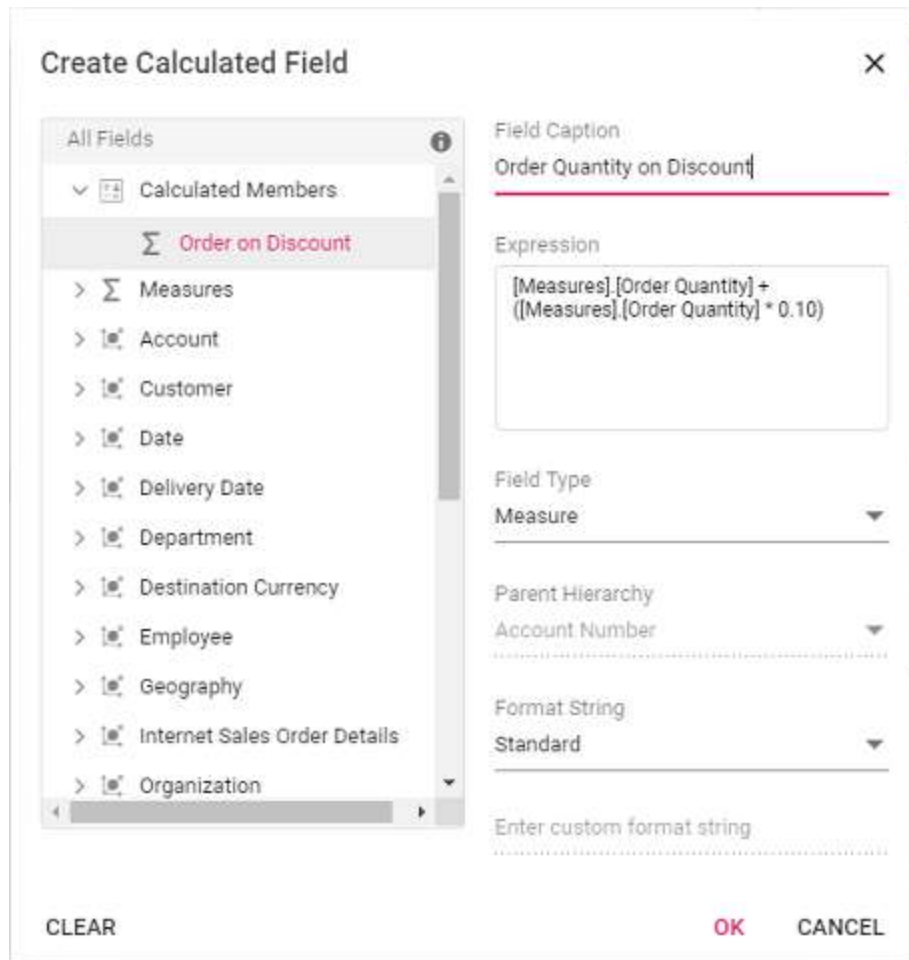
Standard

Enter custom format string

CLEAR

OK

CANCEL



[Editing the existing calculated field formula](#)

Existing calculated field formula can be edited only through the UI at runtime. To do so, open the calculated field dialog, click the target field. User can now see the existing expression getting displayed in a "Expression" section. Now, change the expression based on user requirement and click "OK".

Create Calculated Field

All Fields

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] + (([Measures].[Order Quantity] * 0.10)

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Enter custom format string

CLEAR

OK

CANCEL

The screenshot shows the 'Create Calculated Field' dialog box. On the left, under 'All Fields', the 'Calculated Members' section is expanded, showing a list of fields. The field 'Order on Discount' is selected and highlighted in red. The right side of the dialog contains the following fields:

- Field Caption:** Order Quantity on Discount
- Expression:**
$$[\text{Measures}].[Order\ Quantity] + ([\text{Measures}].[Order\ Quantity] * 0.50)$$
- Field Type:** Measure
- Parent Hierarchy:** Account Number
- Format String:** Standard
- Enter custom format string:** (empty text box)

At the bottom of the dialog, there are three buttons: 'CLEAR', 'OK', and 'CANCEL'.

[Reusing the existing formula in a new calculate field](#)

While creating a new calculated field, if user wants to add the formula of an existing calculated field, it can be done easily. To do so, simply drag-and-drop the existing calculated field to the "Expression" section.

Create Calculated Field

All Fields

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Enter custom format string

CLEAR

OK

CANCEL

Create Calculated Field

All Fields:

- Calculated Members
 - Order on Discount**
- Measures
- Account
- Customer
- Date
- Delivery Date
- Department
- Destination Currency
- Employee
- Geography
- Internet Sales Order Details
- Organization

Field Caption

Discount Quantity

Expression

Example: $[Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)$

Order on Discount

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Enter custom format string

CLEAR

OK

CANCEL

Create Calculated Field

All Fields

- Calculated Members
 - Order on Discount**
- Measures
- Account
- Customer
- Date
- Delivery Date
- Department
- Destination Currency
- Employee
- Geography
- Internet Sales Order Details
- Organization

Field Caption
Discount Quantity

Expression
[Measures].[Order on Discount]

Field Type
Measure

Parent Hierarchy
Account Number

Format String
Standard

Enter custom format string

CLEAR **OK** **CANCEL**

Modifying the existing format string

Existing calculated field's format string can be modified only through the UI at runtime. To do so, open the calculated field dialog and click the target calculated field. User can now see the format string for the existing calculated field getting displayed in a drop-down list. Change the format string based on the requirement and finally click "OK".

Create Calculated Field

All Fields

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)

Field Type

Measure

Parent Hierarchy

Account Number

Format String

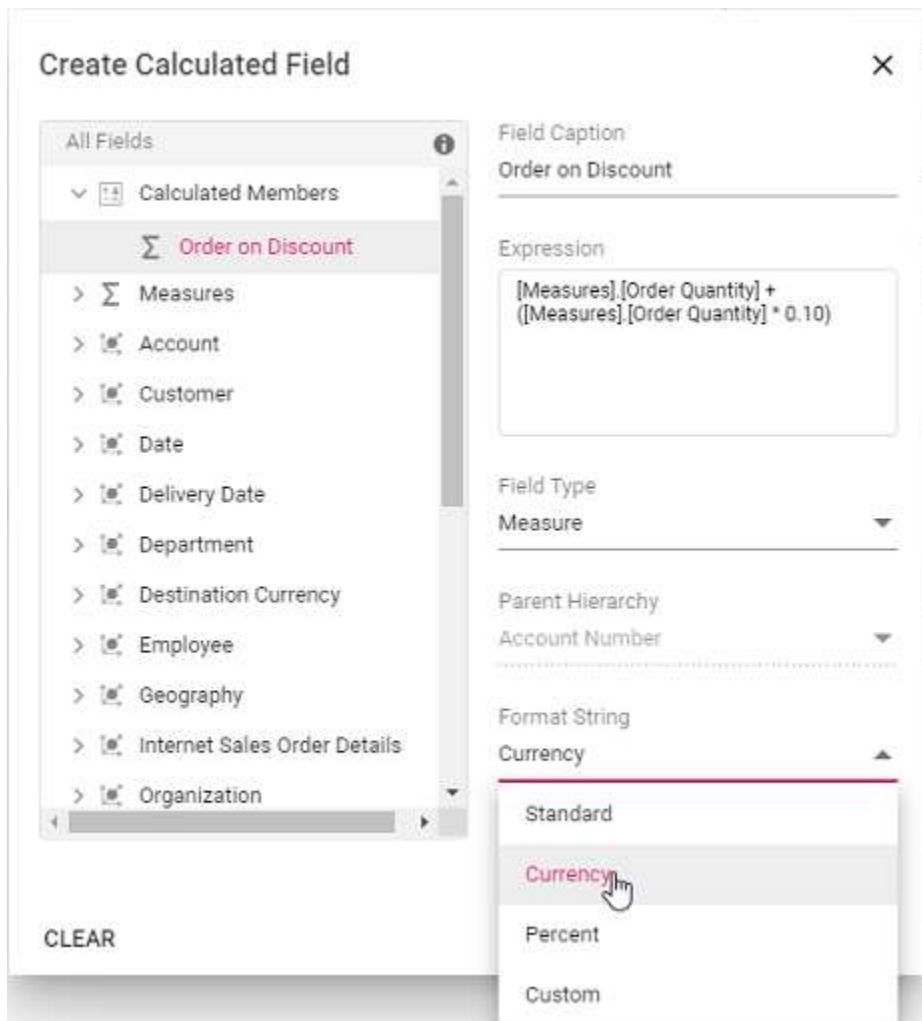
Standard

Enter custom format string

CLEAR

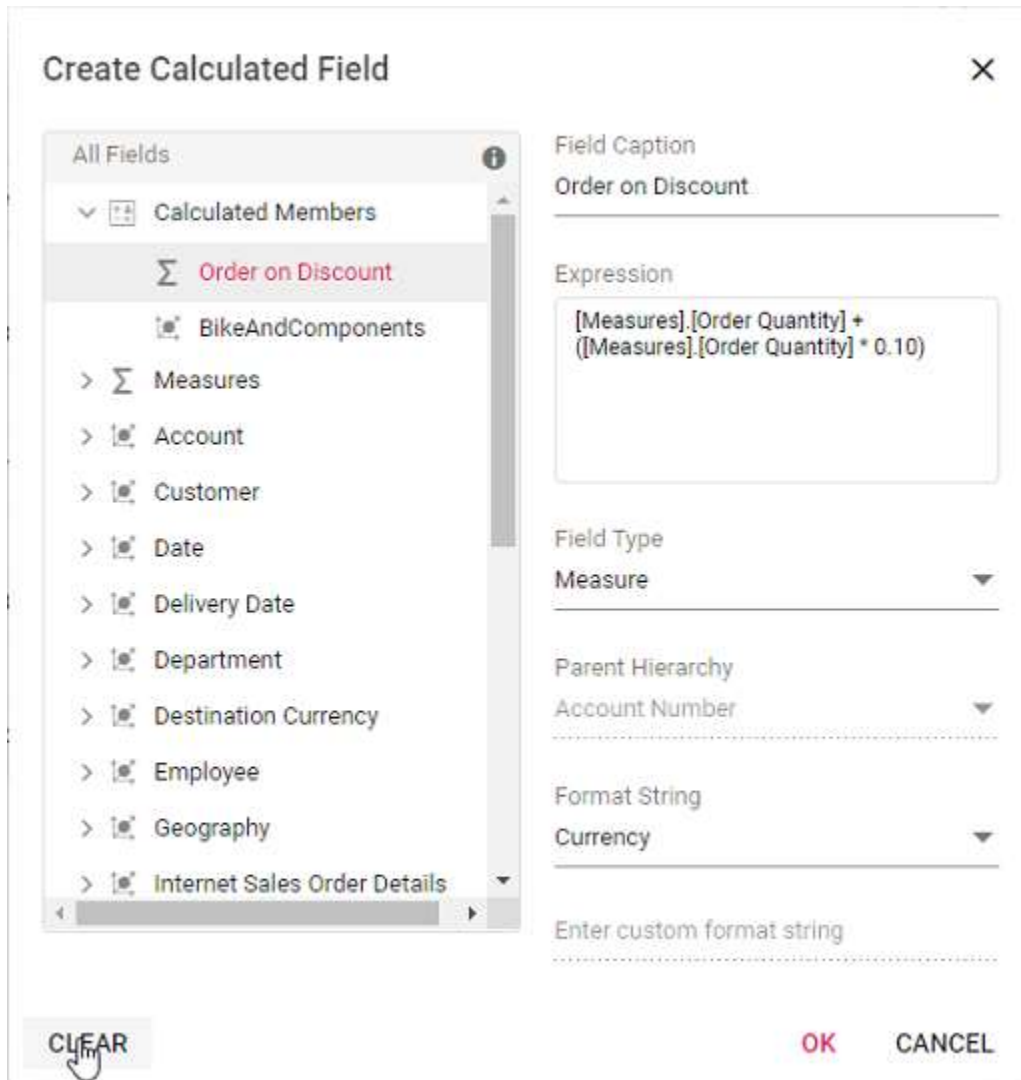
OK

CANCEL



[Clearing the changes while editing the calculated field](#)

Previous changes can be cleared by using the "Clear" option while performing operations such as creating and editing the calculated field. To do so, click the "Clear" button in the bottom left corner of the dialog.



Virtual Scrolling

Allows large amounts of data to be loaded without any performance degradation by rendering rows and columns in relation to the current viewport. Rest of the data will be brought into the viewport dynamically based on vertical or horizontal scroll position. This feature can be enabled by setting the [enableVirtualization](#) property to **true**.

To use the virtual scrolling feature, inject the **VirtualScroll** module into the pivot table.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    catalog: 'Adventure Works DW 2008 SE',
    cube: 'Adventure Works',
    providerType: 'SSAS',
    enableSorting: true,
    url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
    localeIdentifier: 1033,
    rows: [
```

```

        { name: '[Customer].[Customer]', caption: 'Customer
Geography' },
    ],
    columns: [
        { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
        { name: '[Measures]', caption: 'Measures' },
    ],
    values: [
        { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
        { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
    ],
    formatSettings: [{ name: '[Measures].[Internet Sales Amount]',
format: 'C0' }]
    },
    height: 350,
    enableVirtualization: true
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations for virtual scrolling

- The [columnWidth](#) property in [gridSettings](#) should be in pixels. The percentage value is not accepted.
- Resizing columns and setting the width of individual columns will affect scrolling and is therefore not recommended.
- The grand totals option is not supported by virtual scrolling.

Run the application

The quickstart project is configured to compile and run the application in the browser. Use the following command to run the application.

`

npm start

`

Output will be displayed as follows.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        catalog: 'Adventure Works DW 2008 SE',
        cube: 'Adventure Works',
        providerType: 'SSAS',
        enableSorting: true,
        url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
        localeIdentifier: 1033,
        rows: [
            { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },

```

```

        ],
        columns: [
            { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
            { name: '[Measures]', caption: 'Measures' },
        ],
        values: [
            { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
            { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
        ],
        filters: [
            { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
        ],
        calculatedFieldSettings: [
            {
                name: 'BikeAndComponents',
                formula: '([Product].[Product
Categories].[Category].[Bikes] + [Product].[Product
Categories].[Category].[Components] )',
                hierarchyUniqueName: '[Product].[Product Categories]',
                formatString: 'Standard'
            },
            {
                name: 'Order on Discount',
                formula: '[Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)',
                formatString: 'Currency'
            }
        ],
        filterSettings: [
            {
                name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
'[Date].[Fiscal].[Fiscal Year].&[2005]'],
                levelCount: 3
            }
        ]
    },
    showFieldList: true,
    showGroupingBar: true,
    allowCalculatedField: true,
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Deploy the Application

The Essential JS 2 pivot table control features are segregated into individual feature-wise modules. The [Essential Studio JavaScript \(Essential JS 2\)](#) build and [CDN](#) scripts contains code for all features used in pivot table and hence we suggest to not to use them in production. We strongly recommend you to generate script files to use in production using our [Custom Resource Generator \(CRG\)](#) for Essential JS 2. CRG will allow you to generate the bundled script for the currently enabled features in pivot table.

Data Binding

To bind OLAP datasource to the pivot table, you need to specify following properties under [dataSourceSettings](#) option.

Properties	Description
cube	Points the respective cube name from OLAP database.
providerType	Points the provider type for pivot table to identify the type of data source.
url	Contains the cube URL for establishing the connection (online).
catalog	Contains the database name (catalog name) to fetch the data.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    catalog: 'Adventure Works DW 2008 SE',
    cube: 'Adventure Works',
    providerType: 'SSAS',
    enableSorting: true,
    url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
    localeIdentifier: 1033,
    rows: [
      { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
    ],
    columns: [
      { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
      { name: '[Measures]', caption: 'Measures' },
    ],
    values: [
      { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
      { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
    ],
    filters: [
      { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
    ],
    filterSettings: [
      {
        name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
'[Date].[Fiscal].[Fiscal Year].&[2005]'],
        levelCount: 3
      }
    ]
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

*Fields**Measures in row axis*

By default, the measures are plotted in column axis. To plot those measures in row axis, place the **Measures** button in the row axis as follows.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    catalog: 'Adventure Works DW 2008 SE',
    cube: 'Adventure Works',
    providerType: 'SSAS',
    enableSorting: true,
    url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
    localeIdentifier: 1033,
    rows: [
      { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
      { name: '[Measures]', caption: 'Measures' }
    ],
    columns: [
      { name: '[Product].[Product Categories]', caption: 'Product
Categories' }
    ],
    values: [
      { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
      { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
    ],
    filters: [
      { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
    ],
    filterSettings: [
      {
        name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
'[Date].[Fiscal].[Fiscal Year].&[2005]'],
        levelCount: 3
      }
    ]
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Measures in different position

You can place measures in different position in row or column axis either thorough code behind or UI. In this sample, **measures** placed before the dimension in the column axis.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {

```

```

        catalog: 'Adventure Works DW 2008 SE',
        cube: 'Adventure Works',
        providerType: 'SSAS',
        enableSorting: true,
        url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
        localeIdentifier: 1033,
        rows: [
            { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
        ],
        columns: [
            { name: '[Measures]', caption: 'Measures' },
            { name: '[Product].[Product Categories]', caption: 'Product
Categories' }
        ],
        values: [
            { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
            { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
        ],
        filters: [
            { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
        ],
        filterSettings: [
            {
                name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
'[Date].[Fiscal].[Fiscal Year].&[2005]'],
                levelCount: 3
            }
        ],
        height: 350
    });
    pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Named set

Named set is a multidimensional expression (MDX) that returns a set of dimension members, which can be created by combining the cube data, arithmetic operators, numbers, and functions.

You can bind the named sets in the pivot table by setting its unique name in the [name](#) property either in row or column axis and [isNamedSet](#) boolean property to **true**. In this sample, we have added "Core Product Group" named set in the column axis.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        catalog: 'Adventure Works DW 2008 SE',
        cube: 'Adventure Works',
        providerType: 'SSAS',
        enableSorting: true,
        url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
        localeIdentifier: 1033,
    }
});

```

```

        rows: [
            { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
        ],
        columns: [
            {
                name: "[Core Product Group]",
                isNamedSet: true
            },
            { name: '[Measures]', caption: 'Measures' },
        ],
        values: [
            { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
            { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
        ],
        filters: [
            { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
        ],
        filterSettings: [
            {
                name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                levelCount: 3
            }
        ]
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Configuring authentication

Users can configure basic authentication information to access the OLAP cube using the [authentication](#) property. The settings required to configure are as follows:

- [userName](#): It allows the user to set a username that recognizes the basic authentication of the IIS.
- [password](#): It allows to set the appropriate password.

If the user does not configure the authentication, a default popup will appear in the browser to get the authentication information.

`ts

```

var pivotTableObj = new ej.pivotview.PivotView({
dataSourceSettings: {
catalog: 'Adventure Works DW 2008 SE',
cube: 'Adventure Works',
providerType: 'SSAS',

```

```
enableSorting: true,
url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
localeIdentifier: 1033,
rows: [
{ name: '[Customer].[Customer Geography]', caption: 'Customer Geography' },
],
columns: [
{ name: '[Product].[Product Categories]', caption: 'Product Categories' },
{ name: '[Measures]', caption: 'Measures' },
],
values: [
{ name: '[Measures].[Customer Count]', caption: 'Customer Count' },
{ name: '[Measures].[Internet Sales Amount]', caption: 'Internet Sales Amount' }
],
filters: [
{ name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
],
filterSettings: [
{
name: '[Date].[Fiscal]', items: ['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
'[Date].[Fiscal].[Fiscal Year].&[2005]'],
levelCount: 3
}
],
authentication: {
userName: 'username',
password: 'password'
}
},
height: 350
});
pivotTableObj.appendTo('#PivotTable');
`
```


Roles

SQL Server Analysis Services uses [roles](#) to limit data access within a cube. Each role defines a set of permissions that can be granted to a single user or groups of users. It is used to manage security by limiting access to sensitive data and determining who has access to and can change the cube. It can be configured using the [roles](#) property in [dataSourceSettings](#).

The [roles](#) property can be used to specify one or more roles to the OLAP cube, separated by commas.

```
`ts
```

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    catalog: 'Adventure Works DW 2008 SE',
    cube: 'Adventure Works',
    roles: 'Role1',
    providerType: 'SSAS',
    url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
    localeIdentifier: 1033,
    rows: [
      { name: '[Customer].[Customer Geography]', caption: 'Customer Geography' },
    ],
    columns: [
      { name: '[Product].[Product Categories]', caption: 'Product Categories' },
      { name: '[Measures]', caption: 'Measures' },
    ],
    values: [
      { name: '[Measures].[Customer Count]', caption: 'Customer Count' },
      { name: '[Measures].[Internet Sales Amount]', caption: 'Internet Sales Amount' }
    ],
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

OLAP Cube: Elements

Field list

The field list, aka cube dimension browser, is a tree view like structure that organizes the cube elements such as dimensions, hierarchies, measures, etc., from the selected cube into independent logical groups.

Types of node in field list

- **Display folder:** A folder that contains a set of similar elements.
- **Measure:** Quantity available for analysis.
- **Dimension:** A name given to the parts of the cube that categorizes data.
- **Attribute Hierarchy:** Level of attributes down the hierarchy.
- **User-defined Hierarchy:** Members of a dimension in a hierarchical structure.
- **Level:** Denotes a specific level in the category.
- **Named Set:** A collection of tuples and members, that can be defined and saved as a part of cube definition for later use.

Measure

In a cube, a measure is a set of values that are based on a column in the cube's fact table and are usually numeric. The measures are the central values of a cube that are analyzed. That is, measures are the numeric data of primary interest to users browsing a cube. You can select measures depend on the types of users request. Some common measures are sales, costs, expenditures, and production count.

Dimension

A simple dimension object is composed of basic information such as name, hierarchy, level, and members. You can create a dimension element by specifying its name and providing the hierarchy and level name. The dimension element contains the hierarchical details and information about each included level elements in that hierarchy. A hierarchy can have any number of level elements and the level elements can have any number of members and the member elements can have any number of child members.

Hierarchy

Each element of a dimension can be summarized using a hierarchy. The hierarchy is a series of parent-child relationship, where a parent member represents the consolidation of members which are its children. Parent members can be further aggregated as the children of another parent. For example, May 2005 can be summarized into Second Quarter 2005 which in turn would be summarized in the year 2005.

Level

Level element is the child of hierarchy element which contains a set of members, each of which has the same rank within a hierarchy.

Attribute hierarchy

Attribute hierarchy contains the following levels:

- A leaf level contains distinct attribute member, and each member of the leaf level is known as a leaf member.
- Intermediate levels if the attribute hierarchy is a parent-child hierarchy.
- An optional (all) level contains the aggregated value of the attribute hierarchy's leaf members, with the member of the (all) level also known as the (all) member.

User-defined hierarchy

User-defined hierarchy organizes the members of a dimension into hierarchical structure and provides navigation paths in a cube. For example, take a dimension table that supports three attributes such as year, quarter, and month. The year, quarter, and month attributes are used to construct a user-defined hierarchy, named Calendar, in the time dimension that relates to all levels.

Differentiating user-defined hierarchy and attribute hierarchy

- User-defined hierarchy contains more than one level whereas attribute hierarchy contains only one level.
- User-defined hierarchy provides the navigation path between the levels taken from attribute hierarchies of the same dimension.
- The attribute hierarchy and the user-defined hierarchy are represented in different ways as shown in the following table.

Named set

A named set is a collection of tuples and members, which can be defined and saved as a part of the cube definition. Named set records reside inside the sets folder, which is under a dimension element. These elements can be dragged to [rows](#) or [columns](#) axis via grouping bar or field list at runtime. To work with a lengthy, complex, or commonly used expression easier, Multidimensional Expressions (MDX) allows you to define a named set.









Calculated field

The calculated field allows user to insert or add a new calculated field based on the available OLAP cube elements from the bound data source. Calculated fields are nothing but customized dimensions or measures that are newly created based on the user-defined expression.

The two types of calculated fields are as follows:

- **Calculated Measure** – Creates a new measure through user-defined expression.
- **Calculated Dimension** – Creates a new dimension through user-defined expression.

Symbolic representation of the nodes inside field list

Icon	Name	Node type	Is Draggable	
	Display Folder	Display Folder	False	
	Measure	Measure	False	
	Dimension	Dimension	False	
	User Defined Hierarchy	Hierarchy	True	
	Attribute Hierarchy	Hierarchy	True	
	 	Levels (in order)	Level Element	True
	Named Set	Named Set	True	

In general, the Pivot Table is created using the built-in engine for given data source. This is an optional feature that allows you to create the Pivot Table with a server-side pivot engine and external data binding. And this option is applicable only for relational data source.

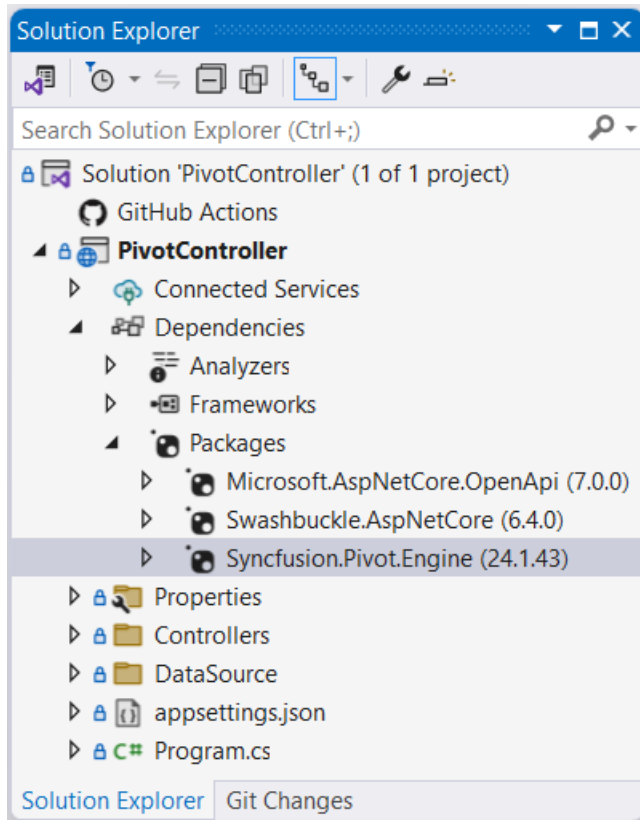
Server side pivot engine in EJ2 JavaScript Pivotview control

This section briefs the Syncfusion assembly [Syncfusion.Pivot.Engine](#), which is used in a server-side application to perform all Pivot calculations such as aggregation, filtering, sorting, grouping, and so on, and only the information to be displayed in the Pivot Table's viewport is passed to the client-side (browser) via web service (Web API) rather than the entire data source. It reduces network traffic and improves the rendering performance of the Pivot Table, especially when dealing with large amounts of data. It also works best with virtual scrolling enabled and supports all the Pivot Table's existing features.

Quick steps to render the Pivot Table by using the server-side Pivot Engine

Download and installing Server-side Pivot Engine

1. Download the ASP.NET Core-based stand-alone Pivot Table [application](#) from the GitHub repository.
2. The **PivotController** (Server-side) application that is downloaded includes the following files.
 - **PivotController.cs** file under **Controllers** folder – This helps to do data communication with Pivot Table.
 - **DataSource.cs** file under **DataSource** folder – This file has model classes to define the structure of the data sources.
 - The sample data source files **sales.csv** and **sales-analysis.json** under **DataSource** folder.
3. Open the **PivotController** application in Visual Studio where the Syncfusion library [Syncfusion.Pivot.Engine](#) will be downloaded automatically from the nuget.org site.



Connecting Pivot Table to Server-side Pivot Engine

1. Run the **PivotController** (Server-side) application which will be hosted in IIS shortly.
2. Then in the Pivot Table sample, set the [mode](#) property under [dataSourceSettings](#) as **Server** and map the URL of the hosted Server-side application in [URL](#) property of [dataSourceSettings](#).

```
`javascript
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    url: 'https://localhost:44350/api/pivot/post',
    mode: 'Server',
    //Other codes here...
  }
});
pivotObj.appendTo('#PivotTable');
```

3. Frame and set the report based on the data source available in the **PivotController** application.

```
`javascript
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
```

```

url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
rows: [{
name: 'ProductID', caption: 'Product ID'
}],
formatSettings: [{
name: 'Price', format: 'C'
}],
columns: [{
name: 'Year', caption: 'Production Year'
}],
values: [
{ name: 'Sold', caption: 'Units Sold' },
{ name: 'Price', caption: 'Sold Amount' }
],
//Other codes here...
});
pivotObj.appendTo('#PivotTable');

```

4. Run the sample to get the following result.

	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
PRO-10	7901220	\$1,841,712.00	7561880	\$1,762,804.00	7591
PRO-11	7174625	\$1,673,533.00	7338175	\$1,711,599.80	7164
PRO-12	9168140	\$2,135,848.80	7689400	\$1,792,704.00	7375
PRO-13	6674280	\$1,560,619.20	8585260	\$2,001,738.40	6571
PRO-14	7489240	\$1,745,841.20	7167840	\$1,675,479.20	7396
PRO-15	8149185	\$1,902,397.00	7450585	\$1,734,093.00	7247
PRO-16	7070150	\$1,652,838.40	8037500	\$1,873,597.60	7883

Available configurations in Server-side application

[Supportive Data Sources](#)

The server-side Pivot Engine supports the following data sources,

- Collection
- JSON

- CSV
- DataTable
- Dynamic

Collection

The collection data sources such as List, IEnumerable, and so on are supported. This can be bound using the **GetData** controller method. Also, in the Pivot Table sample, set the [type](#) property under [dataSourceSettings](#) to **JSON**, which is also the default enumeration value.

In the server-side application (**PivotController**), a collection type data source is framed in the **DataSource.cs** file as shown in the following.

```
`c#
public class PivotViewData
{
    public string ProductID { get; set; }
    public string Country { get; set; }
    public string Product { get; set; }
    public double Sold { get; set; }
    public double Price { get; set; }
    public string Year { get; set; }
    public List<PivotViewData> GetVirtualData()
    {
        List<PivotViewData> VirtualData = new List<PivotViewData>();
        for (int i = 1; i <= 10000; i++)
        {
            PivotViewData p = new PivotViewData
            {
                ProductID = "PRO-" + ((100 + i)%20),
                Year = (new string[] { "FY 2015", "FY 2016", "FY 2017", "FY 2018", "FY 2019" })[new Random().Next(5)],
                Country = (new string[] { "Canada", "France", "Australia", "Germany", "France" })[new Random().Next(5)],
                Product = (new string[] { "Car", "Van", "Bike", "Flight", "Bus" })[new Random().Next(5)],
                Price = (3.4 * i) + 500,
                Sold = (i * 15) + 10
            };
            VirtualData.Add(p);
        }
    }
}
```

```

return VirtualData;
}
}
`

```

To bind the data source, set its model type **PivotViewData** to **TValue** of the **PivotEngine** class.

```

`c#
private PivotEngine<DataSource.PivotViewData> PivotEngine = new
PivotEngine<DataSource.PivotViewData>();
`

```

Then call the data source in **GetData** method of **PivotController.cs** file.

```

`c#
public async Task<object> GetData(FetchData param)
{
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind the collection type data source.
return new DataSource.PivotViewData().GetVirtualData();
});
}
`

```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```

`javascript
var pivotObj = new ej.pivotview.PivotView({
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
type: 'JSON',
rows: [{
name: 'ProductID', caption: 'Product ID'
}],

```



```

formatSettings: [{
  name: 'Price', format: 'C'
}],
columns: [{
  name: 'Year', caption: 'Production Year'
}],
values: [
  { name: 'Sold', caption: 'Units Sold' },
  { name: 'Price', caption: 'Sold Amount' }
],
//Other codes here...
});

```

	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
PRO-10	7901220	\$1,841,712.00	7561880	\$1,762,804.00	7591
PRO-11	7174625	\$1,673,533.00	7338175	\$1,711,599.80	7164
PRO-12	9168140	\$2,135,848.80	7689400	\$1,792,704.00	7375
PRO-13	6674280	\$1,560,619.20	8585260	\$2,001,738.40	6571
PRO-14	7489240	\$1,745,841.20	7167840	\$1,675,479.20	7396
PRO-15	8149185	\$1,902,397.00	7450585	\$1,734,093.00	7247
PRO-16	7070150	\$1,652,838.40	8037500	\$1,873,597.60	7885

JSON

The JSON data from a local *.json file type can be connected to the Pivot Table. Here, the file can be read by the **StreamReader** option, which will give the result in the string format. The resultant string needs to be converted to collect data that can be bound to the Server-side pivot engine.

In the Server-side application, **sales-analysis.json** file is available under **DataSource** folder and its model type is defined in **DataSource.cs** file.

```

`c#
public class PivotJSONData
{
  public string Date { get; set; }
  public string Sector { get; set; }
  public string EnerType { get; set; }
  public string EneSource { get; set; }
}

```

```

public int PowUnits { get; set; }
public int ProCost { get; set; }
public List<PivotJSONData> ReadJSONData(string url)
{
    WebClient myWebClient = new WebClient();
    Stream myStream = myWebClient.OpenRead(url);
    StreamReader stream = new StreamReader(myStream);
    string result = stream.ReadToEnd();
    stream.Close();
    return Newtonsoft.Json.JsonConvert.DeserializeObject<List<PivotJSONData>>(result);
}
}
`

```

To bind the data source, set its model type **PivotJSONData** to **TValue** of the **PivotEngine** class.

```

`c#
private PivotEngine<DataSource.PivotJSONData> PivotEngine = new PivotEngine<DataSource.
PivotJSONData>();
`

```

Then call the data source in **GetData** method of **PivotController.cs** file.

```

`c#
public async Task<object> GetData(FetchData param)
{
    return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
    async (cacheEntry) =>
    {
        cacheEntry.SetSize(1);
        cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
        // Here bind JSON type data source from the sales-analysis.json file.
        return new DataSource.PivotJSONData().ReadJSONData(_hostingEnvironment.ContentRootPath +
        "/DataSource/sales-analysis.json");
    });
}
`

```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```

`javascript
var pivotObj = new ej.pivotview.PivotView({
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
type: 'JSON',
rows: [{
name: 'EneSource', caption: 'Energy Source'
}],
formatSettings: [{
name: 'ProCost', format: 'C'
}],
columns: [{
name: 'EnerType', caption: 'Energy Type'
}],
values: [
{ name: 'PowUnits', caption: 'Units Sold' },
{ name: 'ProCost', caption: 'Sold Amount' }
],
},
//Other codes here...
});
`

```

	Biomass		Free Energy		Grand Total
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
Bio-diesel	1042	\$1,439.00			1042
Ethanol Fuel	595	\$1,031.00			595
Geo-thermal			1528	\$2,115.00	1528
Hydro-electric			3378	\$3,244.00	3378
Solar			7929	\$6,210.00	7929
Wastage	712	\$1,043.00			712
Wind			15571	\$7,666.00	15571

JSON data from any remote server, like a local JSON file, can also be supported. It accepts both directly downloadable files (*.json*) and web service URLs. To bind this, the URL of the *.json* file of a remote server

has to be mapped under the **GetData** method. The rest of the configurations are the same as described above.

In the server-side application, the CDN link is used to connect the same **sales-analysis.json** file which is already hosted in the Syncfusion server.

```
`c#
public async Task<object> GetData(FetchData param)
{
    return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
    async (cacheEntry) =>
    {
        cacheEntry.SetSize(1);
        cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
        // Here bind JSON type data source from remote server.
        return new DataSource.PivotJSONData().ReadJSONData("http://cdn.syncfusion.com/data/sales-
        analysis.json");
    });
}
```

CSV

The CSV data from a local *.csv file type can be connected to the Pivot Table. Here, the file can be read by the **StreamReader** option, which will give the result in the string format. The resultant string needs to be converted to collect data that can be bound to the server-side pivot engine. Also, in the Pivot Table sample, set the [type](#) property under [dataSourceSettings](#) as **CSV**.

In the server application, the **sales.csv** file is available under the **DataSource** folder, and its model type is defined in the **DataSource.cs** file.

```
`c#
public class PivotCSVData
{
    public string Region { get; set; }
    public string Country { get; set; }
    public string ItemType { get; set; }
    public string SalesChannel { get; set; }
    public string OrderPriority { get; set; }
    public string OrderDate { get; set; }
    public int OrderID { get; set; }
    public string ShipDate { get; set; }
```

```

public int UnitsSold { get; set; }
public double UnitPrice { get; set; }
public double UnitCost { get; set; }
public double TotalRevenue { get; set; }
public double TotalCost { get; set; }
public double TotalProfit { get; set; }
public List<string[]> ReadCSVData(string url)
{
    List<string[]> data = new List<string[]>();
    using (StreamReader reader = new StreamReader(new WebClient().OpenRead(url)))
    {
        string line;
        while ((line = reader.ReadLine()) != null)
        {
            line = line.Trim();
            if (!string.IsNullOrEmpty(line))
            {
                data.Add(line.Split(','));
            }
        }
        return data;
    }
}

```

To bind the data source, set its model type **PivotCSVData** to **TValue** of the **PivotEngine** class.

```

`c#
private PivotEngine<DataSource.PivotCSVData> PivotEngine = new PivotEngine<DataSource.
PivotCSVData>();

```

Then call the data source in **GetData** method of **PivotController.cs** file.

```

`c#
public async Task<object> GetData(FetchData param)

```

```

{
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind CSV type data source from sales.csv file.
return new DataSource.PivotCSVData().ReadCSVData(_hostingEnvironment.ContentRootPath +
"//DataSource//sales.csv");
});
}
`

```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```

`javascript
var pivotObj = new ej.pivotview.PivotView({
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
type: 'CSV',
rows: [{
name: 'ItemType', caption: 'Item Type'
}],
formatSettings: [{
name: 'UnitPrice', format: 'C'
}],
columns: [{
name: 'Region'
}],
values: [
{ name: 'UnitsSold', caption: 'Units Sold' },
{ name: 'UnitPrice', caption: 'Sold Amount' }
],
},
}

```

```
//Other codes here...
```

```
});
```

```
,
```

	Asia		Australia and Oceania		Central Ar
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
Baby Food	55810	\$3,318.64	21548	\$765.84	
Beverages	84400	\$806.65	34486	\$332.15	
Cereal	32668	\$1,439.90	43195	\$1,645.60	
Clothes	24290	\$764.96	17020	\$655.68	
Cosmetics	84630	\$7,432.40	35220	\$3,060.40	
Fruits	35679	\$74.64	29951	\$46.65	
Household	28166	\$4,009.62	55444	\$6,014.43	

CSV data from any remote server, like a local CSV file, can also be supported. It accepts both directly downloadable files (.csv) and web service URLs. To bind this, the URL of the .csv file of a remote server has to be mapped under **GetData** method. The rest of the configurations are the same as described above.

In the server application, the CDN link is used to connect the same **sales.csv** file which is already hosted in the Syncfusion server.

```
`c#
```

```
public async Task<object> GetData(FetchData param)
{
    return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
    async (cacheEntry) =>
    {
        cacheEntry.SetSize(1);
        cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
        // Here bind CSV type data source from remote server.
        return new DataSource.PivotCSVData().ReadCSVData("http://cdn.syncfusion.com/data/sales-
analysis.csv");
    });
}
```

DataTable

In the server-side application, there is a manually created DataTable **BusinessObjectsDataView** by mapping the model type **PivotViewData** in **DataSource.cs** file.

```
`c#
```

```

public class BusinessObjectsDataView
{
    public DataTable GetDataTable()
    {
        DataTable dt = new DataTable("BusinessObjectsDataTable");
        PropertyDescriptorCollection pdc = TypeDescriptor.GetProperties(typeof(PivotViewData));
        foreach (PropertyDescriptor pd in pdc)
        {
            dt.Columns.Add(new DataColumn(pd.Name, pd.PropertyType));
        }
        List<PivotViewData> list = new PivotViewData().GetVirtualData();
        foreach (PivotViewData bo in list)
        {
            DataRow dr = dt.NewRow();
            foreach (PropertyDescriptor pd in pdc)
            {
                dr[pd.Name] = pd.GetValue(bo);
            }
            dt.Rows.Add(dr);
        }
        return dt;
    }
}

```

To bind the data source, set its model type **PivotViewData** to **TValue** of the **PivotEngine** class.

```

`c#
private PivotEngine<DataSource.PivotViewData> PivotEngine = new
PivotEngine<DataSource.PivotViewData>();

```

Then call the data source in **GetData** method of **PivotController.cs** file.

```

`c#
public async Task<object> GetData(FetchData param)
{

```



```
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind the DataTable.
return new DataSource.BusinessObjectsDataView().GetDataTable();
});
}
```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```
`javascript
var pivotObj = new ej.pivotview.PivotView({
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
rows: [{
name: 'ProductID', caption: 'Product ID'
}],
formatSettings: [{
name: 'Price', format: 'C'
}],
columns: [{
name: 'Year', caption: 'Production Year'
}],
values: [
{ name: 'Sold', caption: 'Units Sold' },
{ name: 'Price', caption: 'Sold Amount' }
],
}
//Other codes here...
});
```

	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
PRO-10	7901220	\$1,841,712.00	7561880	\$1,762,804.00	759
PRO-11	7174625	\$1,673,533.00	7338175	\$1,711,599.80	716
PRO-12	9168140	\$2,135,848.80	7689400	\$1,792,704.00	737
PRO-13	6674280	\$1,560,619.20	8585260	\$2,001,738.40	657
PRO-14	7489240	\$1,745,841.20	7167840	\$1,675,479.20	739
PRO-15	8149185	\$1,902,397.00	7450585	\$1,734,093.00	724
PRO-16	7070150	\$1,652,838.40	8037500	\$1,873,597.60	788

Dynamic

The model type has to be defined in the aforementioned data sources. However, there is no need to define a model type for the following data sources, which are also supported by the server-side pivot engine.

ExpandoObject

In the server-side application, an **ExpandoObject** type data source is available under the class **PivotExpandoData** in **DataSource.cs** file.

```
`c#
public class PivotExpandoData
{
    public List<ExpandoObject> Orders { get; set; } = new List<ExpandoObject>();
    public List<ExpandoObject> GetExpandoData()
    {
        Orders = Enumerable.Range(1, 75).Select((x) =>
        {
            dynamic d = new ExpandoObject();
            d.OrderID = 1000 + (x % 100);
            d.CustomerID = (new string[] { "ALFKI", "ANANTR", "ANTON", "BLONP", "BOLID" })[new Random().Next(5)];
            d.Freight = (new double[] { 2, 1, 4, 5, 3 })[new Random().Next(5)] * x;
            d.OrderDate = (new DateTime[] { new DateTime(2010, 11, 5), new DateTime(2018, 10, 3), new DateTime(1995, 9, 9), new DateTime(2012, 8, 2), new DateTime(2015, 4, 11) })[new Random().Next(5)];
            d.ShipCountry = (new string[] { "USA", "UK" })[new Random().Next(2)];
            d.Verified = (new bool[] { true, false })[new Random().Next(2)];
            return d;
        }).Cast<ExpandoObject>().ToList<ExpandoObject>();
        return Orders;
    }
}
```

```

}
}
`

```

To bind the data source, set its model type as **ExpandoObject** to **TValue** of the **PivotEngine** class.

```

`c#
private PivotEngine<ExpandoObject> PivotEngine = new PivotEngine<ExpandoObject>();
`

```

Then call the data source in **GetData** method of **PivotController.cs** file.

```

`c#
public async Task<object> GetData(FetchData param)
{
    return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
    async (cacheEntry) =>
    {
        cacheEntry.SetSize(1);
        cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
        // Here returns ExpandoObject type data source.
        return new DataSource.PivotExpandoData().GetExpandoData();
    });
}
`

```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```

`javascript
var pivotObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        url: 'https://localhost:44350/api/pivot/post',
        mode: 'Server',
        rows: [{
            name: 'CustomerID', caption: 'Customer ID'
        }],
        columns: [{
            name: 'ShipCountry', caption: 'Ship Country'
        }],
    },
});

```

```

values: [
{ name: 'Freight', caption: 'Units Sold' }
]
}
//Others codes here...
});
`

```

	UK	USA	Grand Total
	Units Sold	Units Sold	Units Sold
ALFKI	1187	738	1925
ANANTR	1011	814	1825
ANTON	1046	632	1678
BLONP	1080	749	1829
BOLID	492	929	1421
Grand Total	4816	3862	8678

Dynamic Objects

In the server-side application, a data source is framed by dynamic objects which is available under the class **PivotDynamicData** in the **DataSource.cs** file.

```

`c#
public class PivotDynamicData
{
    public List<DynamicDictionary> Orders = new List<DynamicDictionary>() { };
    public List<DynamicDictionary> GetDynamicData()
    {
        Orders = Enumerable.Range(1, 100).Select((x) =>
        {
            dynamic d = new DynamicDictionary();
            d.OrderID = 100 + x;
            d.CustomerID = (new string[] { "ALFKI", "ANANTR", "ANTON", "BLONP", "BOLID" })[new
            Random().Next(5)];
            d.Freight = (new double[] { 2, 1, 4, 5, 3 })[new Random().Next(5)] * x;
            d.OrderDate = (new DateTime[] { new DateTime(2010, 11, 5), new DateTime(2018, 10, 3), new
            DateTime(1995, 9, 9), new DateTime(2012, 8, 2), new DateTime(2015, 4, 11) })[new Random().Next(5)];
            d.ShipCountry = (new string[] { "USA", "UK" })[new Random().Next(2)];
            d.Verified = (new bool[] { true, false })[new Random().Next(2)];
            return d;
        }
    }
}

```

```

}).Cast<DynamicDictionary>().ToList<DynamicDictionary>());
return Orders;
}
public class DynamicDictionary : System.Dynamic.DynamicObject
{
    Dictionary<string, object> dictionary = new Dictionary<string, object>();
    public override bool TryGetMember(GetMemberBinder binder, out object result)
    {
        string name = binder.Name;
        return dictionary.TryGetValue(name, out result);
    }
    public override bool TrySetMember(SetMemberBinder binder, object value)
    {
        dictionary[binder.Name] = value;
        return true;
    }
    //The "GetDynamicMemberNames" method of the "DynamicDictionary" class must be overridden and
    return the property names to perform data operation and editing while using dynamic objects.
    public override System.Collections.Generic.IEnumerable<string> GetDynamicMemberNames()
    {
        return this.dictionary?.Keys;
    }
}
}
}
`

```

To bind the data source, set its class **PivotDynamicData** to **TValue** of the **PivotEngine** class.

```

`c#
private PivotEngine<DataSource.PivotDynamicData> PivotEngine = new
PivotEngine<DataSource.PivotDynamicData>();
`

```

Then call the data source in **GetData** method of **PivotController.cs** file.

```

`c#
public async Task<object> GetData(FetchData param)
{

```

```
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind data source with dynamic objects.
return new DataSource.PivotDynamicData().GetDynamicData();
});
}
```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```
`javascript
var pivotObj = new ej.pivotview.PivotView({
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
rows: [{
name: 'CustomerID', caption: 'Customer ID'
}],
columns: [{
name: 'ShipCountry', caption: 'Ship Country'
}],
values: [
{ name: 'Freight', caption: 'Units Sold' }
]
}
//Other codes here...
});
```

	UK	USA	Grand Total
	Units Sold	Units Sold	Units Sold
ALFKI	1030	742	1772
ANANTR	1352	1018	2370
ANTON	2525	1782	4307
BLONP	1786	1002	2788
BOLID	1409	2012	3421
Grand Total	8102	6556	14658

Controller Configuration

Memory Cache

In the server-side application, the [Memory Cache](#) option is used to store the data source and engine properties in RAM, which will be used for UI operations. To improve performance, this limits the execution of all initial rendering code to regenerate the aggregated values during each UI operation. The codes below show how we use the memory cache option in the **GetEngine** method to store engine properties.

```
`c#
public async Task<EngineProperties> GetEngine(FetchData param)
{
    isRendered = false;
    // Engine properties are stored in memory cache with GUID "param.Hash".
    return await _cache.GetOrCreateAsync("engine" + param.Hash,
    async (cacheEntry) =>
    {
        isRendered = true;
        cacheEntry.SetSize(1);
        // Memory cache expiration time can be set here.
        cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
        PivotEngine.Data = await GetData(param);
        return await PivotEngine.GetEngine(param);
    });
}
```

The engine properties are stored in RAM as a cache with a unique ID (GUID) that is transferred from the client-side source code. The GUID is generated at random and will be changed if the page containing the Pivot Table is refreshed or opened in a new tab/window. As a result, each GUID's memory cache contains unique information, and the component operates independently.

Meanwhile, the memory cache is set to expire after 60 minutes from RAM to free its memory. If the component is still running, the data will be generated and stored for another 60 minutes.

Methods and its needs

- **Post:** Allows to get the information from the client-side source and calls appropriate controller methods.
- **GetEngine:** Allows to store the engine properties in RAM as a cache which fires on initial rendering or when the memory cache is expired.
- **GetData:** Allows to store data source in RAM as a cache which fires on initial rendering or when the memory cache is expired.
- **GetMembers:** Allows to get the members of a field. This fires when the member editor is opened to do a filtering operation.
- **GetRawData:** Allows to get raw data of an aggregated value cell. This fires when the drill-through or editing dialog is opened.
- **GetPivotValues:** Allows to update the stored engine properties in-memory cache and returns the aggregated values to browser to render the Pivot Table. Here, the return value can be modified. The Pivot Table will be rendered browser-based on this.

Pivot chart in EJ2 JavaScript Pivotview control

In pivot table component, pivot chart would act as an additional visualization component with its basic and important characteristic like drill down and drill up, 15+ chart types, series customization, axis customization, legend customization, export, print and tooltip. Its main purpose is to show the pivot data in graphical format.

If user prefers, the pivot chart component can also be displayed individually with pivot values and can change the report dynamically with the help of field list and grouping bar. Using the `displayOption` property in pivot table, user can set the visibility of grid and chart in pivot table component. It holds below properties,

- **view:** Specifies the pivot table component to display grid alone or chart alone or both.
- **primary:** Specifies the pivot table to display either grid or chart as primary component during initial loading. It is applicable only when setting the property **view** to **Both**.

The below sample displays the pivot chart component based on the pivot report bound on it.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: { chartSeries: { type: 'Column' } }
});
```



```
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Data binding

End user can bind both local and remote data binding options available in the component to feed the data. The [dataSource](#) property can be assigned either with an instance of **DataManager** or JavaScript object array collection.

For more information [refer](#) here.

Chart Types

Supports 21 different types of charts as follows,

- Line
- Column
- Area
- Bar
- StepArea
- StackingLine
- StackingColumn
- StackingArea
- StackingBar
- StepLine
- Pareto
- Bubble
- Scatter
- Spline
- SplineArea
- StackingLine100
- StackingColumn100
- StackingBar100
- StackingArea100
- Polar
- Radar

Line is the default pivot chart type. User can change the pivot chart type by using the property [type](#) in [chartSeries](#).

In the below code sample, the pivot chart type is set as **Bar**.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
```

```

        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Chart' },
    chartSettings: { chartSeries: { type: 'Bar' } }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>

```

```

    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accumulation Charts

Supports 4 different types of accumulation charts as follows,

- Pie
- Doughnut
- Funnel
- Pyramid

As like other chart types it can be changed using the property [type](#) in [chartSeries](#).

In the below code sample, the **Pie** chart is rendered, and the other accumulation charts can be switched using the drop-down list.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year' }, { name: 'Products' }],
        rows: [{ name: 'Country' }, { name: 'Quarter' }],
        formatSettings: [{ name: 'Amount', format: 'C' }],
        values: [{ name: 'Amount' }, { name: 'Sold' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Chart' },
    chartSettings: { chartSeries: { type: 'Pie' } }
});
pivotTableObj.appendTo('#PivotTable');
var chartTypesDropDown = new ej.dropdowns.DropDownList({
    floatLabelType: 'Auto',
    change: function (args) {
        pivotTableObj.chartSettings.chartSeries.type = args.value;
    }
});
chartTypesDropDown.appendTo('#charttypes');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drill Down/Up

In the accumulation charts, drill down and drill up operations can be performed using the built-in context menu option. It will be shown while clicking on the chart series. The context menu has the following options:

Expand - It is to drill down the corresponding series until the last level.

Collapse - It is to drill up the corresponding series until the first level.

Exit - It is to close the context menu.

The drill operation in accumulation charts can be performed only for row headers.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    columns: [{ name: 'Products' }],
    rows: [{ name: 'Country' }, { name: 'Year' }, { name: 'Quarter' },],
    formatSettings: [{ name: 'Amount', format: 'C' }],
    values: [{ name: 'Amount' }, { name: 'Sold' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: { chartSeries: { type: 'Pie' } }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

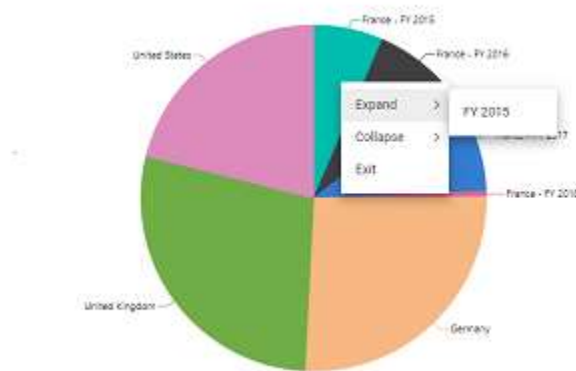
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```



Column Headers and Delimiters

Unlike other chart types, the accumulation charts consider the values of a single column from the pivot table to be drawn. Preferably the first column of the pivot table is considered by default. But it can be changed by defining the column headers using the `columnHeader` property in [chartSettings](#).

If the column has more than one header, then need to mention all the headers separated by the delimiter -, for example, **Germany-Road Bikes**. Using the property `columnDelimiter` in [chartSettings](#), one can set the desired delimiter to separate the column headers.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['Germany'] }],
        enableSorting: true,
        columns: [{ name: 'Country' }, { name: 'Products' }],
        rows: [{ name: 'Year' }, { name: 'Quarter' }],
        formatSettings: [{ name: 'Amount', format: 'C' }],
        values: [{ name: 'Amount' }, { name: 'Sold' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Chart' },

```

```

    chartSettings: { columnHeader: 'Germany-Road Bikes', columnDelimiter: '-'
    }, chartSeries: { type: 'Doughnut' } }
  });
  pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div class="container">
    <div id="PivotTable"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```


Label Customization

The data labels are visible by default showing header name. Its visibility can be modified using the `visible` boolean property in `dataLabel`. With regard to the label arrangement, the **Smart Labels** options help to arrange labels efficiently without overlapping. It can be disabled by setting the `enableSmartLabels` property in `chartSettings` as `false`.

The `position` property in `dataLabel` allows to specify the position of the data label. The available options are,

- **Outside**: Positions the label outside the point. It is the default option.
- **Inside**: Positions the label inside the point.

In the following code sample, the data labels are placed inside.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    columns: [{ name: 'Year' }, { name: 'Products' }],
    rows: [{ name: 'Country' }, { name: 'Quarter' }],
    formatSettings: [{ name: 'Amount', format: 'C' }],
    values: [{ name: 'Amount' }, { name: 'Sold' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: {
    enableSmartLabels: false,
    chartSeries: { dataLabel: { visible: true, position: 'Inside' },
  type: 'Pyramid' }
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The **Connector Line** will be visible when the data label is placed outside the chart. It can be customized using the `connectorStyle` property in `dataLabel` for its color, length, width etc. In the following code sample, the connector line is customized.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year' }, { name: 'Products' }],
        rows: [{ name: 'Country' }, { name: 'Quarter' }],
        formatSettings: [{ name: 'Amount', format: 'C' }],
        values: [{ name: 'Amount' }, { name: 'Sold' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Chart' },
    chartSettings: {
        chartSeries: {

```

```

        dataLabel: {visible: true, position: 'Outside', connectorStyle:
{ length: '50px', width: 2, dashArray: '5,3', color: '#f4429e' } },
        type: 'Funnel'
    }
}
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Pie and Doughnut Customization

User can draw pie and doughnut charts within the specified range using the `startAngle` and `endAngle` properties in `chartSeries`. The default value of the `startAngle` property is **0**, and the `endAngle` property is **360**. By customizing these properties, user can draw semi pie and semi doughnut charts.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    columns: [{ name: 'Year' }, { name: 'Products' }],
    rows: [{ name: 'Country' }, { name: 'Quarter' }],
    formatSettings: [{ name: 'Amount', format: 'C' }],
    values: [{ name: 'Amount' }, { name: 'Sold' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: { chartSeries: { startAngle: 270, endAngle: 90, type:
'Doughnut' } }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Users can get doughnut chart from pie chart and vice-versa using the `innerRadius` property in [chartSeries](#). If the property is greater than 0 percent, the doughnut chart will appear from the pie chart.

It takes the value only in percentage.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year' }, { name: 'Products' }],
        rows: [{ name: 'Country' }, { name: 'Quarter' }],
        formatSettings: [{ name: 'Amount', format: 'C' }],
        values: [{ name: 'Amount' }, { name: 'Sold' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Chart' },
    chartSettings: { chartSeries: { startAngle: 270, endAngle: 90, type:
'Doughnut' } }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exploding Series Points

Exploding can be enabled by setting the `explode` property in [chartSeries](#) to **true**. The series points will be exploded either on mouse click or touch.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year' }, { name: 'Products' }],

```

```

        rows: [{ name: 'Country' }, { name: 'Quarter' }],
        formatSettings: [{ name: 'Amount', format: 'C' }],
        values: [{ name: 'Amount' }, { name: 'Sold' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Chart' },
    chartSettings: { chartSeries: { explode: true, type: 'Pie' } }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div class="container">
    <div id="PivotTable"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Field List

User can enable the field list by setting the property [showFieldList](#) in pivot table as **true**. By using this, user can customize the report dynamically and view the result in pivot chart. For more information regarding the field list, refer the [field list](#) topic.

In the following sample, the **Popup** mode of field list is enabled in the pivot chart integration.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: { chartSeries: { type: 'Column' }},
  showFieldList: true
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouping Bar

User can enable the grouping bar by setting the property [showGroupingBar](#) in pivot table to **true**. The grouping bar in pivot chart shows a dropdown list in value axis instead of buttons. The dropdown list holds list of value fields bounded in the [dataSourceSettings](#) and it can be switched to draw the pivot chart with the selected value field. This has been defined as the default behavior in the pivot chart component. For more information regarding the grouping bar, refer the [grouping bar](#) topic.

For multiple axis support, buttons will be placed in value axis instead of dropdown list.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],

```

```

        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 240,
    displayOption: { view: 'Chart' },
    chartSettings: { chartSeries: { type: 'Column' }},
    showGroupingBar: true
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>

```

```

        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

For accumulation charts alone, a drop-down list will be placed in the column axis instead of the buttons. The drop-down list shows the column headers available in the pivot table. Users can dynamically switch column headers with the help of the drop-down list, and the accumulation chart will be updated accordingly.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year' }, { name: 'Products' }],
        rows: [{ name: 'Country' }, { name: 'Quarter' }],
        formatSettings: [{ name: 'Amount', format: 'C' }],
        values: [{ name: 'Amount' }, { name: 'Sold' }],
        filters: []
    },
    displayOption: { view: 'Chart' },
    chartSettings: { chartSeries: { type: 'Pie' } },
    showGroupingBar: true,
    height: 240
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Single Axis

By default, the pivot chart will be drawn with the value field (measure) which is set first in the report under value axis. But, user can change to specific value field using the property [value](#) in [chartSettings](#).

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Chart' },
    chartSettings: { value: 'Amount', chartSeries: { type: 'Column' } }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple Axis

User can draw the pivot chart with multiple value fields by setting the property [enableMultipleAxis](#) in [chartSettings](#) to **true**. In the below code sample, the pivot chart will be drawn with both value fields "Sold" and "Amount" available in the [dataSourceSettings](#).

The multiple axis support is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: { enableMultipleAxis: true, chartSeries: { type: 'Column'
} }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If the user binds more value fields, the result will be multiple pivot charts, and each chart will shrink within the parent container height. To avoid this, set the [enableScrollOnMultiAxis](#) property in [chartSettings](#) to **true**. By doing so, each pivot chart will only shrink to a minimal “160px” - “180px” height showing a vertical scrollbar for a clear view.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }, { name: 'Products', type: 'Count' }],
        rows: [{ name: 'Country' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 550,
    displayOption: { view: 'Chart' },
    chartSettings: { enableMultipleAxis: true,
enableScrollOnMultiAxis: true,
chartSeries: { type: 'Column' } }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
```



```
</body></html>
```

Meanwhile, there is another way to display multiple values in a chart. In this approach, the series drawn from multiple values are grouped and displayed in a single chart. And, based on the values, multiple Y axis scales will be framed with different ranges. This can be achieved by setting the properties [enableMultipleAxis](#) as **true** and [multipleAxisMode](#) as **Single** in [chartSettings](#).

In the following code sample, the pivot chart can be seen as a single chart with multiple value fields such as **Sold** and **Amount** that are drawn as multiple Y axis.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: { enableMultipleAxis: true, multipleAxisMode : 'Single',
chartSeries: { type: 'Column' } }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Additionally, to display chart series for multiple values within a single y-axis, set the properties [enableMultipleAxis](#) to **true** and the [multipleAxisMode](#) to **Combined**, in the [chartSettings](#).

The y-axis range values will be formatted using the first value field on the value axis. For example, if the first value field is in currency format and the remaining value fields are in different number formats or no format, the y-axis range values will be displayed in the currency format of the first value field.

The pivot chart in the following code sample can be seen as a single chart with multiple value fields such as **Sold** and **Amount** drawn as a single y-axis.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],

```

```

        filters: []
    },
    height: 350,
    showGroupingBar: true,
    showFieldList: true,
    displayOption: { view: 'Chart' },
    chartSettings: { enableMultipleAxis: true, multipleAxisMode :
'Combined', chartSeries: { type: 'Column' } }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>

```

```

        <div id="PivotTable"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show point color based on members

When multiple axes are enabled, you can display the same color for each member in the column axis by setting the [showPointColorByMembers](#) property to **true** in the [chartSettings](#). As a result, the end user can easily identify each member across different measures in the entire chart.

Furthermore, end user can see or hide specific members across different measures in the entire chart with a single click on the legend item.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Chart' },
    chartSettings: {
        title: 'Sales Analysis', value: 'Amount', chartSeries: { type:
'Column' },
        enableMultipleAxis: true, showPointColorByMembers: true,
        multipleAxisMode: 'Stacked',
        primaryYAxis: {border: {width: '0'}}
    },
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Series Customization

User can customize series of the pivot chart using [chartSeries](#) in [chartSettings](#). The changes handled in the property will be reflected commonly in all chart series.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
```

```

    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Chart' },
    chartSettings: { chartSeries: { type: 'Column', enableTooltip: false,
border: { color: '#000', width: 2 } } }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can also customize the pivot chart series individually using the [chartSeriesCreated](#) event, which occurs after the pivot chart series has been created. You can customize each series individually by iterating them.

In the following sample, the even series are hidden in the pivot chart.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Chart' },
    chartSeriesCreated: (args) => {
        for (var pos = 0; pos < args.series.length; pos++) {
            if (pos % 2 == 0) {
                args.series[pos].visible = false;
            }
        }
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Display chart series points in desired color palettes

By default, the pivot chart displays each data point in the chart series with a pre-defined color combination. And, depending on the number of data points drawn in each chart series, this color

combination will change automatically. You can, however, design your own color scheme by using the [palettes](#) property in the [chartSettings](#).

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: { chartSeries: { type: 'Column' }, palettes: ["#E94649",
"#F6B53F", "#6FAAB0", "#C4C24A" ] },
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">
```

```

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis Customization

User can customize axis of the pivot chart using [primaryXAxis](#) and [primaryYAxis](#) properties in [chartSettings](#).

Axis customization is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

In the following sample, title of y-axis and x-axis are customized.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Chart' },
    chartSettings: {
        chartSeries: { type: 'Column' },
        primaryXAxis: { title: 'X axis title' },
        primaryYAxis: { title: 'Y axis title' }
    }
});

```

```
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

One can also customize multi-level labels of primary x-axis by using the [multiLevelLabelRender](#) event in the [chartSettings](#), which fires on rendering each multi-level label in the pivot chart. It has the following parameters:

axis - It holds the information of the current axis.

text - It allows to change the text of the multi-level label.

textStyle - It allows to customize the text font.

alignment - It allows to set the text alignment.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: {
    chartSeries: { type: 'Column' },
    multiLevelLabelRender(args) {
      args.alignment = 'Near';
      args.textStyle = { fontFamily: 'Bold', fontWeight: '400', size:
'16px', color: 'red' };
      if (args.text === ' + United Kingdom') {
        args.text = 'Text Changed';
        args.textStyle = { fontFamily: 'Bold', fontWeight: '800',
size: '16px', color: 'Blue' };
      }
    }
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Customization

User can customize legend using [legendSettings](#) in [chartSettings](#). By default, legend will be visible and it can be hidden by setting the property [visible](#) in [legendSettings](#) to **false**.

The pivot chart support different types of legend shapes as follows,

- Circle

- Rectangle
- VerticalLine
- Pentagon
- InvertedTriangle
- SeriesType
- Triangle
- Diamond
- Cross
- HorizontalLine

Here **SeriesType** would act as the default shape and it can be changed using the property [legendShape](#) in [chartSeries](#).

Also user can set the position of the legend in pivot chart using the property [position](#) in [legendSettings](#). The available options to set the legend position are as follows,

- Auto: Places the legend based on area type. This is the default.
- Top: Displays the legend at the top of the pivot chart.
- Left: Displays the legend at the left of the pivot chart.
- Bottom: Displays the legend at the bottom of the pivot chart.
- Right: Displays the legend at the right of the pivot chart.
- Custom: Displays the legend based on the given x and y values.

By default, the legend is not visible for the accumulation chart types like pie, doughnut, pyramid, and funnel.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: {
    legendSettings: { position: 'Right' },
    chartSeries: { type: 'Column', legendShape: 'Pentagon' }
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

User Interaction

Marker and CrossHair

User can enable and customize the marker and crosshair using [markerSettings](#) and [crosshairSettings](#) properties in [chartSettings](#) respectively.

Also user can enable and customize the crosshair tooltip for axes using [crosshairTooltip](#) in primary X and Y axes

Marker and crosshair is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: {
    crosshair: { enable: true },
    chartSeries: {
      type: 'Line',
      marker: { fill: '#EEE', height: 10, width: 10, shape: 'Pentagon',
visible: true }
    },
    primaryXAxis: { crosshairTooltip: { enable: true, fill: '#ff0000' }
},
    primaryYAxis: { crosshairTooltip: { enable: true, fill: '#0000FF' }
}
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Zooming and Panning

User can customize zooming and panning option using the property [zoomSettings](#) in [chartSettings](#).

The pivot chart support four types of zooming which can be set as follows,

- [EnablePinchZooming](#)
- [EnableSelectionZooming](#)
- [EnableDeferredZooming](#)
- [EnableMouseWheelZooming](#)

and three modes of zooming direction that specifies whether to zoom vertically or horizontally or in both ways which are,

- x: Pivot chart can be zoomed horizontally.
- y: Pivot chart can be zoomed vertically.
- x,y: Pivot chart can be zoomed both vertically and horizontally.

This can be set using the property [mode](#) in [zoomSettings](#). By default, if the pivot chart is zoomed, a toolbar would display with the options - Zoom, ZoomIn, ZoomOut, Pan, Reset. User can also customize its option using the property [toolbarItems](#) in [zoomSettings](#).

Zooming and panning is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

In the following sample, all the four types of zooming are enabled with toolbar options.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: {
    chartSeries: {
      type: 'Column'
    },
    zoomSettings: {
      enableDeferredZooming: true,
      enableMouseWheelZooming: true,
      enablePinchZooming: true,
      enableSelectionZooming: true
    }
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip

By default, tooltip for the pivot chart is enabled. User can customize it by using the property [tooltip](#) in [chartSettings](#).

The tooltip can be disabled by setting the property [enable](#) in [tooltip](#) as **false**.

In the following sample, the default appearance of tooltip can be modified.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Chart' },
  chartSettings: {
    chartSeries: {
      type: 'Column'
    },
    tooltip: {
      enableMarker: true,
      textStyle: { color: '#000' },
      fill: '#FFF',
      opacity: 1,
      border: { color: '#000' }
    }
  }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export

The pivot chart can be exported using the `chartExport` method which holds parameters like export type, file name, PDF orientation, width, and height in the same order. The mandatory parameters for this method are export type and file name whereas other parameters are optional.

The following are the four export types:

- PNG
- JPEG
- SVG
- PDF

In the following code sample, exporting can be done using an external button named as "Export".

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],

```

```

        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    displayOption: { view: 'Chart' },
    chartSettings: {
        chartSeries: {
            type: 'Column'
        }
    }
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#chartexport');
document.getElementById('chartexport').onclick = function () {
    pivotTableObj.chartExport('PNG', 'result');
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="chartexport" type="button" value="Export" name="Export">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print

The rendered pivot chart can be printed directly from the browser by calling `printChart` method.

In the following code sample, printing can be done using an external button named as "Print".

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name: 'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount', caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    displayOption: { view: 'Chart' },
    chartSettings: {
        chartSeries: {
            type: 'Column'
        }
    }
});
pivotTableObj.appendTo('#PivotTable');
var printBtn = new ej.buttons.Button({ isPrimary: true });
printBtn.appendTo('#chartprint');
document.getElementById('chartprint').onclick = function () {
    pivotTableObj.printChart();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Pivot Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="chartprint" type="button" value="Print" name="Print">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drill down in EJ2 JavaScript Pivotview control

Drill down and drill up

The drill down and drill up action helps to view the bound data in detailed and abstract view respectively. By default, if member(s) has children, then expand and collapse icon will be displayed in

the respective row/column header. On clicking the icon, expand or collapse action will be performed automatically through built-in source code. Meanwhile, leaf member(s) does not contain expand and collapse icon.

	► FY 2015		► FY 2016		► FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
▼ France	450	\$714,955	526	\$1,542,104	
Mountain Bikes	197	\$335,688	238	\$405,552	
Road Bikes	253	\$379,267	288	\$1,136,552	
► Germany	440	\$563,515	496	\$1,772,104	
► United States	546	\$754,515	636	\$2,263,104	
Grand Total	1436	\$2,032,985	1658	\$5,577,312	1

Drill position

Allows to drill only the current position of the selected member and exclude the drilled data of selected member in other positions. For example, if "FY 2015" and "FY 2016" have "Quarter 1" member as child in next level, and when end user attempts to drill "Quarter 1" under "FY 2016", only it will be expanded and not "Quarter 1" under "FY 2015".

This feature is built-in and occurs every time when expand or collapse action is done for better performance.

	▼ FY 2015		▼ FY 2016		
	► Q1	FY 2015 Tot...	Q1		
France	114	114	Mountain Bi...	Road Bikes	Q1 Total
Germany	74	74			
United States	144	144			
Grand Total	332	332	181	221	

Expand all

This property is applicable only for the relational data source.

Allows to either expand or collapse all headers that are displayed in row and column axes. To display all headers in expanded state, set the property [expandAll](#) to **true** and to collapse all headers, set the property [expandAll](#) to **false**. By default, [expandAll](#) property is set to **false**.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
```

```

        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: [],
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>

```

```

        <div id="PivotTable"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Expand all headers for specific fields

This property is applicable only for the relational data source.

Allows to expand or collapse all headers for specific fields (only) in row and column axes. To expand headers for a specific field in row or column axis, set the property [expandAll](#) in [rows](#) or [columns](#) to **true**. By default, [expandAll](#) property in [rows](#) or [columns](#) is set to **false**.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' , expandAll:
true }, { name: 'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' , expandAll: true }, { name: 'Products' }],
        filters: [],
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Expand all except specific member(s)

This option is applicable only for the relational data source.

In addition to the previous topic, there is an enhancement to expand all headers except specific header(s) and similarly to collapse all headers except specific header(s). To achieve this, [drilledMembers](#) is used. The required properties of the [drilledMembers](#) are explained below:

- [name](#): It allows to set the field name whose member(s) needs to be specifically drilled.
- [items](#): It allows to set the exact member(s) which needs to be drilled.

The [drilledMembers](#) option always works in vice-versa with respect to the property [expandAll](#) in pivot table. For example, if [expandAll](#) is set to **true**, then the member(s) added in [items](#) collection alone will be in collapsed state.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: [],
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Expand specific member(s)

End user can also manually expand or collapse specific member(s) in each fields under row and column axes using the [drilledMembers](#) from code behind. The required properties of the [drilledMembers](#) are explained below:

- [name](#): It allows to set the field name whose member(s) needs to be specifically drilled.
- [items](#): It allows to set the exact member(s) which needs to be drilled.
- [delimiter](#): It allows to separate next level of member from its parent member.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        drilledMembers: [{ name: 'Quarter', delimiter: '~', items: ['FY
2015~Q1'] }, { name: 'Year', items: ['FY 2015', 'FY 2016'] }],
        rows: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }, { name: 'Products' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        columns: [{ name: 'Country' }],
        filters: [],
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Event

The event [drill](#) triggers every time when a field is expanded or collapsed. For instance using this event user can alter delimiter and drill action for the respective item. It has the following parameters:

- **drillInfo** - It holds the current drilled item information.
- **pivotview** - It holds pivot table instance.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
    drilledMembers: [{ name: 'Quarter', delimiter: '~', items: ['FY
2015~Q1'] }, { name: 'Year', items: ['FY 2015', 'FY 2016'] }],
    rows: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }, { name: 'Products' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    columns: [{ name: 'Country' }],
    filters: [],
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```



```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ActionBegin

The event [actionBegin](#) triggers when the UI actions such as drill down and drill up begin. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action began. The following are the UI actions and their names:

Action	Action Name
Expand	Drill down
Collapse	Drill up

- **cancel**: It allows user to restrict the current action.

In the below sample, drill down and drill up action can be restricted by setting the **args.cancel** option to **true** in the **actionBegin** event.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        allowLabelFilter: true,
        allowValueFilter: true,
        columns: [{ name: 'Year', caption: 'Production Year' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],

```

```

        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    showFieldList: true,
    actionBegin: function (args) {
        if (args.actionName == 'Drill down' || args.actionName == 'Drill
up') {
            args.cancel = true;
        }
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div class="container">
    <div id="PivotTable"></div>

```

```

    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ActionComplete

The event [actionComplete](#) triggers when a UI action such as drill down or drill up, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. The following are the UI actions and their names:

| Action | Action Name |

|-----|-----|

| [Expand](#) | Drill down |

| [Collapse](#) | Drill up |

- **actionInfo**: It holds the unique information about the current UI action. For example, if drill down action is completed, the event argument contains information such as field name and the drill information.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        allowLabelFilter: true,
        allowValueFilter: true,
        columns: [{ name: 'Year', caption: 'Production Year' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    showFieldList: true,
    actionComplete: function (args) {
        if (args.actionName == 'Drill down' || args.actionName == 'Drill
up') {

```

```

        // Triggers when the drill operations are completed.
    }
}
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div class="container">
    <div id="PivotTable"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

ActionFailure

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- **actionName**: It holds the name of the current action failed. The following are the UI actions and their names:

Action	Action Name
----- -----	
Expand	Drill down
Collapse	Drill up

- **errorInfo**: It holds the error information of the current UI action.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  showGroupingBar: true,
  showFieldList: true,
  allowCalculatedField: true,
  showToolbar: true,
  displayOption: { view: 'Both' },
  toolbar: ['New', 'Save', 'Rename', 'Remove', 'Load',
    'Grid', 'Chart', 'MDX', 'Export', 'SubTotal', 'GrandTotal',
    'ConditionalFormatting', 'FieldList'],
  allowExcelExport: true,
  allowConditionalFormatting: true,
  allowPdfExport: true,
  actionFailure: function (args) {
    if (args.actionName == 'Drill down' || args.actionName == 'Drill
up') {
      // Triggers when the current UI action fails to achieve the
desired result.
    }
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div class="container">
    <div id="PivotTable"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Data Shaping**Aggregation in EJ2 JavaScript Pivotview control**

This feature is applicable only for the relational data source.

End user can perform calculations over a group of values (exclusively for value fields bound in value axis) using the aggregation option. By default, values are added (summed) together. The other aggregation types are explained below.

The fields with data type such as number support all aggregation types mentioned below except for **“CalculatedField”**. The fields with data type such as string, date, datetime, boolean, etc., support **“Count”** and **“DistinctCount”** aggregation types alone.

Operator	Description
-----	-----
Sum	Displays the pivot table values with sum.
Product	Displays the pivot table values with product.
Count	Displays the pivot table values with count.
DistinctCount	Displays the pivot table values with distinct count.
Min	Displays the pivot table with minimum value.
Max	Displays the pivot table with maximum value.
Avg	Displays the pivot table values with average.
Median	Displays the pivot table values with median.
Index	Displays the pivot table values with index.
PopulationStDev	Displays the pivot table values with standard deviation of population.
SampleStDev	Displays the pivot table values with sample standard deviation.
PopulationVar	Displays the pivot table values with variance of population.
SampleVar	Displays the pivot table values with sample variance.
RunningTotals	Displays the pivot table values with running totals.
DifferenceFrom	Displays the pivot table values with difference from the value of the base item in the base field.
PercentageOfDifferenceFrom	Displays the pivot table values with percentage difference from the value of the base item in the base field.
PercentageOfGrandTotal	Displays the pivot table values with percentage of grand total of all values.
PercentageOfColumnTotal	Displays the pivot table values in each column with percentage of total values for the column.
PercentageOfRowTotal	Displays the pivot table values in each row with percentage of total values for the row.
PercentageOfParentTotal	Displays the pivot table values with percentage of total of all values based on selected field.
PercentageOfParentColumnTotal	Displays the pivot table values with percentage of its parent total in each column.
PercentageOfParentRowTotal	Displays the pivot table values with percentage of its parent total in each row.

| CalculatedField | Displays the pivot table with calculated field values. It allows user to create a new calculated field alone. |

Assigning aggregation type for value fields through API

For each value field, the aggregation type can be set using the property **type** for each value fields through code-behind. Meanwhile, aggregation types like **DifferenceFrom** and **PercentageOfDifferenceFrom** can check for specific field of specific item using **baseField** and **baseItem** properties. Likewise, **PercentageOfParentTotal** type can for specific field using **baseField** property. For instance, the aggregation type **DifferenceFrom** would intake the specified field and its corresponding member as input and its value is compared across other members in the same field and also across different fields to formulate an appropriate output value.

It can be configured using **type** option for each value fields through code-behind. The settings required for summarize the value fields at initial rendering are:

- **type**: It allows to set the aggregate type of the field.
- **baseField**: It allows to set the base field to aggregate the values.
- **baseItem**: It allows to set the base item to aggregate the values.

By default, the aggregation will be considered as **Sum** to the value fields which had number type and for the value fields which had non-number type such as string, date, datetime, boolean, etc., the aggregation type will be considered as **Count**. **DifferenceFrom** and **PercentageOfDifferenceFrom** can check for specific field of the specific item using **baseField** and **baseItem**. We can consider the **PercentageOfParentTotal** for specific field using **baseField**.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount', type: 'Sum' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: []
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

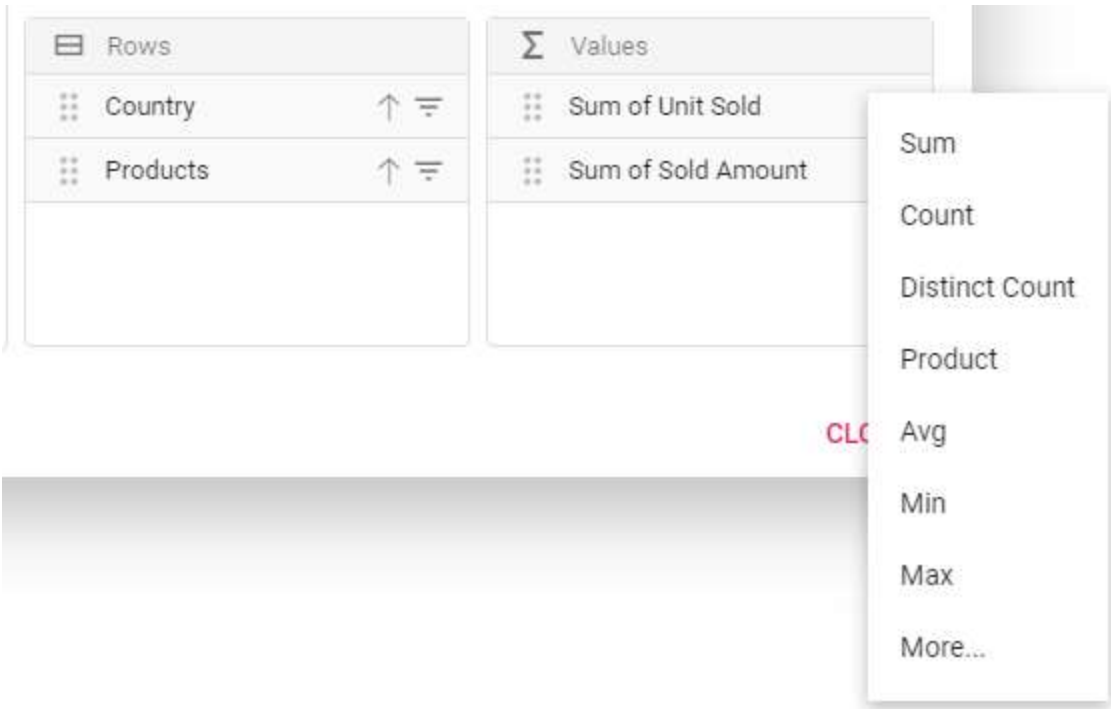
    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

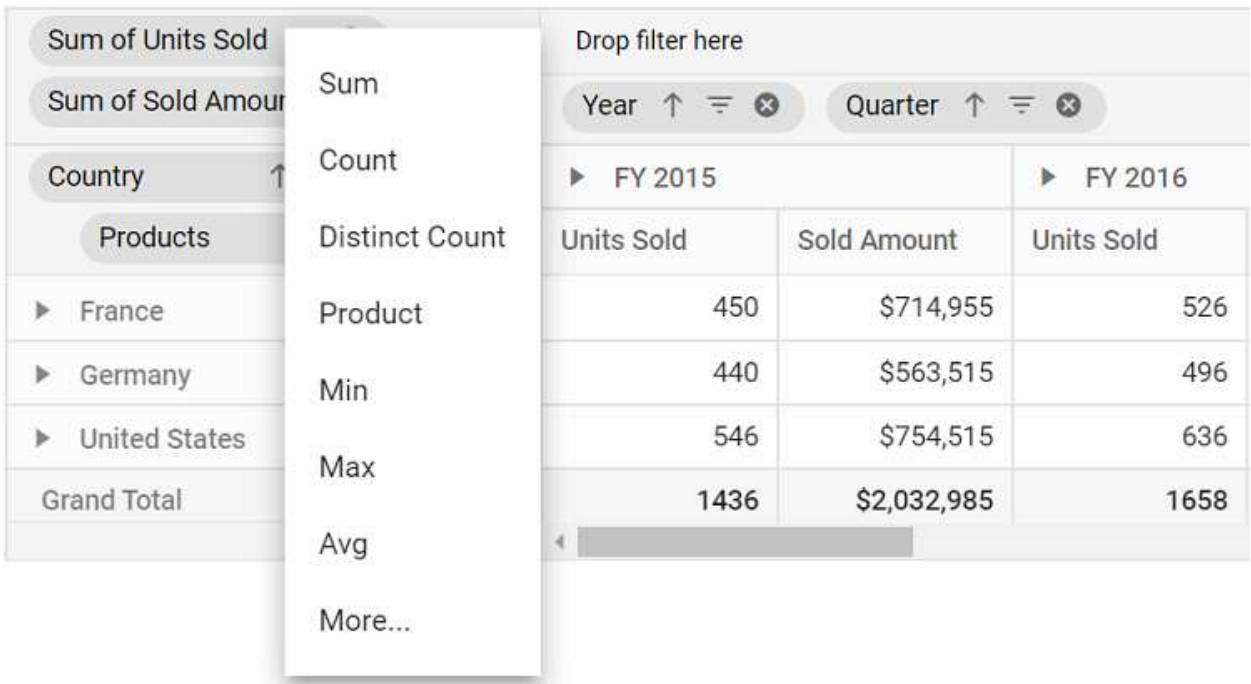
```

Modifying aggregation type for value fields at runtime

Aggregation types can be changed easily through UI at runtime. The value fields bound to grouping bar and field list appears with a dropdown icon which helps to select an appropriate aggregation type for the respective value field. On selection, the values in the pivot table will be changed dynamically.

<!-- markdownlint-disable MD012 -->





Show desired aggregation types in its dropdown menu

By default, all the aggregation types are displayed in the dropdown menu available in buttons. However, based on the request for an application, we may need to show selective aggregation types on our own. This can be achieved using the [aggregateTypes](#) property.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount', type: 'Sum' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: [],
  },
  showFieldList: true,
  showGroupingBar: true,
  height: 350,
  aggregateTypes: ['DistinctCount', 'Avg', 'Product'],
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hiding aggregation type from button text

By default, in value axis each field would be displayed by its name and aggregation type together. To hide aggregation type and display field name alone, set the property [showAggregationOnValueField](#) in [dataSourceSettings](#) to **false**.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount', type: 'Sum' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: [],
        showAggregationOnValueField: false
    },
    showFieldList: true,
    showGroupingBar: true,
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hiding aggregation type icon from UI

By default, the icon to set aggregation type is enabled in the grouping bar. To disable this icon, set the property [showValueTypeIcon](#) in [groupingBarSettings](#) to **false**.

Icon to change the aggregation type can be hidden only in Grouping Bar but not in Field List at the moment.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount', type: 'Sum' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: [],
    showAggregationOnValueField: false
  },
  showGroupingBar: true,
  height: 350,
  groupingBarSettings: {
    showValueTypeIcon: false
  },
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Event

AggregateCellInfo

The event [aggregateCellInfo](#) triggers every time while rendering each value cell. This allows user to change the cell value and skip formatting if applied. It has following parameters:

- **fieldName** - It holds current cell's field name.
- **row** - It holds current cell's row value.
- **column** - It holds current cell's row value.
- **value** - It holds value of current cell.
- **cellSets** - It holds raw data for the aggregated value cell.
- **rowCellType** - It holds row cell type value.
- **columnCellType** - It holds column cell type value.
- **aggregateType** - It holds aggregate type of the cell.
- **skipFormatting** - boolean property, it allows to skip formatting if applied.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount', type: 'Sum' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: [],
        showAggregationOnValueField: false
    },

```

```

        showGroupingBar:true,
        height: 350,
        groupingBarSettings: {
            showValueTypeIcon: false
        },
    });
    pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>

```



```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

ActionBegin

The event [actionBegin](#) triggers when clicking and selecting the aggregate type via the dropdown icon in the value field button, which is present in both grouping bar and field list UI. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action began. For example, while performing aggregation, the action name will be shown as **Aggregate field**.
- **fieldInfo**: It holds the selected value field information.

Note: This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **cancel**: It allows user to restrict the current action.

In the following example, action taken during aggregation type selection via dropdown icon can be restricted by setting the **args.cancel** option to **true** in the **actionBegin** event.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    showGroupingBar: true,
    showFieldList: true,
    actionBegin: function (args) {
        if (args.actionName == 'Aggregate field') {
            args.cancel = true;
        }
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div class="container">
    <div id="PivotTable"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

ActionComplete

The event [actionComplete](#) triggers when a UI action, such as applying aggregation using the dropdown icon via the value field button, which is present in both the grouping bar and the field list UI, is

completed. This allows user to identify the current UI action being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. For example, after completing the aggregation, the action name will be shown as **Field aggregated**.
- **fieldInfo**: It holds the selected value field information.

Note: This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  showGroupingBar: true,
  showFieldList: true,
  actionComplete: function (args) {
    if (args.actionName == 'Aggregate field') {
      // Triggers when the aggregation type is applied.
    }
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ActionFailure

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- **actionName**: It holds the name of the current action failed. For example, if the action fails while performing the aggregation, then the action name will be shown as **Aggregate field**.
- **errorInfo**: It holds the error information of the current UI action.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],

```

```

        filters: []
    },
    showGroupingBar: true,
    showFieldList: true,
    allowCalculatedField: true,
    showToolbar: true,
    displayOption: { view: 'Both' },
    toolbar: ['New', 'Save', 'Rename', 'Remove', 'Load',
        'Grid', 'Chart', 'MDX', 'Export', 'SubTotal', 'GrandTotal',
        'ConditionalFormatting', 'FieldList'],
    allowExcelExport: true,
    allowConditionalFormatting: true,
    allowPdfExport: true,
    actionFailure: function (args) {
        if (args.actionName == 'Aggregate field') {
            // Triggers when the current UI action fails to achieve the
            desired result.
        }
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>

```

```
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Calculated field in EJ2 JavaScript Pivotview control

Allows end user to create a new calculated field in the pivot table, based on available fields from the bound data source or using simple formula with basic arithmetic operators. It can be added at runtime through the built-in dialog, invoked from Field List UI. To do so, set the [allowCalculatedField](#) property to **true** in the pivot table. End user can now see a "CALCULATED FIELD" button enabled in Field List UI automatically, which on clicking will invoke the calculated field dialog and perform necessary operation.

Calculated field can also be included in the pivot table through code behind using the [calculatedFieldsSettings](#). The required properties to create a new calculate field are:

- [name](#): It allows to indicate the calculated field with a unique name.
- [formula](#): It allows to set the formula.
- [formatSettings](#): It helps to set the number format for the resultant value.

The calculated field is applicable only for value fields.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }, { name: 'Total', caption: 'Total Amount', type:
'CalculatedField' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }, { name: 'Total',
format: 'C2' }],
        filters: [],
        calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount) "+"Sum(Sold) "' }],
    },
```

```

        showFieldList: true,
        height: 350,
        allowCalculatedField: true
    });
    pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Meanwhile, user can also view calculated field dialog in UI by invoking [createCalculatedFieldDialog](#) method on an external button click which is shown in the below code sample.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }, { name: 'Total', caption: 'Total Amount', type:
'CalculatedField' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }, { name: 'Total',
format: 'C2' }],
        filters: [],
        calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Sold)"' }]
    },
    showFieldList: true,
    height: 350,
    allowCalculatedField: true
});
pivotTableObj.appendTo('#PivotTable');
var btn = new ej.buttons.Button({ isPrimary: true });
btn.appendTo('#CalculatedField');
document.getElementById('CalculatedField').addEventListener('click', () => {
    pivotTableObj.calculatedFieldModule.createCalculatedFieldDialog();
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

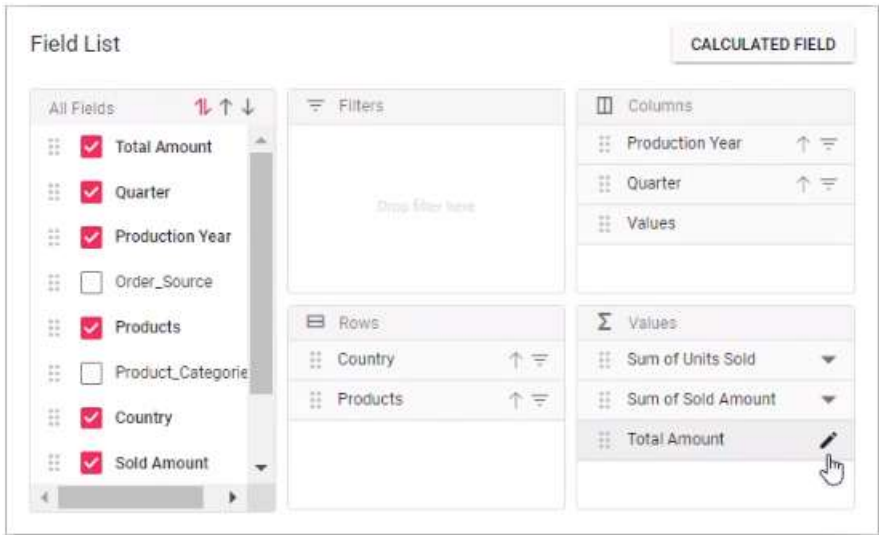
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

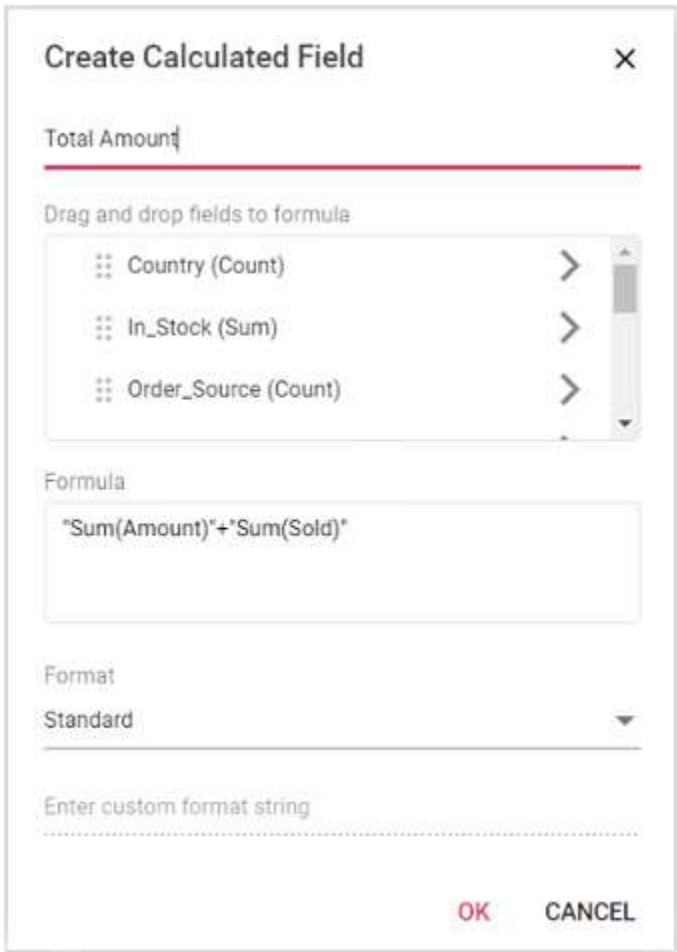
    <div id="container">
        <div id="CalculatedField">Calculated Field</div>
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Editing through the field list and the grouping bar

User can also modify the existing calculated field using the built-in edit option available directly in the field list (or) grouping bar. To do so, click the "Edit" icon available in the calculated field button. Now the calculated field dialog is opened and the current calculated field name, formula and format can be changed at runtime.





Renaming the existing calculated field

Existing calculated field can be renamed only through the UI at runtime. To do so, open the calculated field dialog, select the target field and click "Edit" icon. User can now see the existing name getting displayed in the text box at the top of the dialog. Now, change the name based on user requirement and click "OK".

<!-- markdownlint-disable MD012 -->

Create Calculated Field [X]

Enter the field name

Drag and drop fields to formula

- Sold Amount (Sum)
- Total Amount (Calculated Field)** [Edit]
- Units Sold (Sum)

Formula

Example: ("Sum(Order_Count)" + "Sum(In_Stock)") * 250

Format

None

Enter custom format string

OK CANCEL

Create Calculated Field [X]

Total Sales Amount

Drag and drop fields to formula

- Sold Amount (Sum) >
- Total Amount (Calculated Field)** [trash] [edit]
- Units Sold (Sum) >

Formula

"Sum(Amount)"+"Sum(Sold)"

Format

Standard

Enter custom format string

OK CANCEL

Editing the existing calculated field formula

Existing calculated field formula can be edited only through the UI at runtime. To do so, open the calculated field dialog, select the target field and click "Edit" icon. User can now see the existing formula getting displayed in a multiline text box at the bottom of the dialog. Now, change the formula based on user requirement and click "OK".

Create Calculated Field

×

Enter the field name

Drag and drop fields to formula

Sold Amount (Sum)

>

Total Amount (Calculated Field)

✖

✎

Units Sold (Sum)

>

Formula

Example: ("Sum(Order_Count)" + "Sum(In_Stock)") * 250

Format

None

Enter custom format string

OK

CANCEL

Create Calculated Field

×

Total Amount

Drag and drop fields to formula

Sold Amount (Sum)

Total Amount (Calculated Field)

Units Sold (Sum)

Formula

"Sum(Amount)"+"Sum(Sold)"

Format

Standard

Enter custom format string

OK CANCEL

Reusing the existing formula in a new calculate field



While creating a new calculated field, if user wants to add the formula of an existing calculated field, it can be done easily. To do so, simply drag-and-drop the existing calculated field to the "Formula" section.

Create Calculated Field ×

Sales Amount

Drag and drop fields to formula

Sold Amount (Sum) >

Total Amount (Calculated Field)  

Total Amount (Calculated Field) >

Formula

Example: ("Sum(Order_Count)" + "Sum(In_Stock)") * 250

Format

None ▼

Enter custom format string

OK CANCEL



Create Calculated Field

×

Sales Amount

Drag and drop fields to formula

Sold Amount (Sum) >

Total Amount (Calculated Field)  

Units Sold (Sum) >

Formula

Example: (=Sum(Order_Count) + 'Sum(In_Stock)') * 250

Total Amount (Calculated Field)

Format

None

Enter custom format string

OK CANCEL

Create Calculated Field

Sales Amount

Drag and drop fields to formula

- Sold Amount (Sum)
- Total Amount (Calculated Field)
- Units Sold (Sum)

Formula

Sum(Amount)'+Sum(Sold)

Format

None

Enter custom format string

OK CANCEL

Apply the format to the calculated field values

Values in a new or existing calculated field can be formatted via the calculated field UI or code behind. The [formatSettings](#) property in code-behind can be used to specify the desired format. For more information about the supported formats refer [here](#).

To apply format to calculated field values at runtime via UI, a built-in dropdown under the "Format" label is available, from which the user can select the pre-defined format options listed below.

- **Standard** - Denotes the numeric type.
- **Currency** - Denotes the currency type.
- **Percent** - Denotes the percentage type.
- **Custom** - Denotes the custom format. For example: "C2". This shows the value "9584.3" as "\$9584.30."
- **None** - Denotes that no format will be applied.

By default, **None** will be selected from the dropdown.

The screenshot shows the 'Create Calculated Field' dialog box. At the top, the title is 'Create Calculated Field' with a close button (X). Below the title, the name of the calculated field is 'Total Amount'. Under the heading 'Drag and drop fields to formula', there is a list of fields: 'Sold Amount (Sum)', 'Total Amount (Calculated Field)' (which is highlighted in red and has a trash icon and a pencil icon), and 'Units Sold (Sum)'. Below this list is a 'Formula' text box containing the formula `"Sum(Amount)*Sum(Sold)"`. At the bottom, there is a 'Format' section with a dropdown menu. The dropdown is open, showing the following options: 'Standard' (which is highlighted in red and has a mouse cursor pointing at it), 'Currency', 'Percent', 'Custom', and 'None'.

In addition, you can specify the desired custom formats by selecting the **Custom** option from the "Format" dropdown.

Create Calculated Field

×

Sales Amount

Drag and drop fields to formula

Sold Amount (Sum)

Total Amount (Calculated Field)

Units Sold (Sum)

Formula

"Sum(Amount)"+"Sum(Sold)"

Format

Custom

C2

I

OK

CANCEL

Supported operators and functions for the calculated field formula

Below is a list of operators and functions that can be used in the formula to create the calculated fields.

- **+** – addition operator.

`ts

Syntax: X + Y

,

- **-** – subtraction operator.

`ts

Syntax: X - Y

,

- `*` – multiplication operator.

`ts

Syntax: $X * Y$

`

- `/` – division operator.

`ts

Syntax: X / Y

`

- `^` – power operator.

Grouping in EJ2 JavaScript Pivotview control

This feature is applicable only for the relational data source.

Grouping is the most-useful feature in pivot table and the component automatically groups date, time, number and string. For example, the date type can be formatted and displayed based on year, quarter, month, and more. Likewise, the number type can be grouped range-wise, such as 1-5, 6-10, etc. These group fields will act as individual fields and allows users to drag them between different axes such as columns, rows, values, and filters and create pivot table at runtime.

<!-- markdownlint-disable MD012 -->

Filtering in EJ2 JavaScript Pivotview control

Filtering allows to view the pivot table with selective records based on members that can be either included or excluded through UI and code-behind.

The following are the three different types of filtering:

- Member filtering
- Label filtering
- Value filtering

When all the above filtering options are disabled via code-behind, then the filter icon would be disabled in the field list or grouping bar UI.

Member filtering

Sorting in EJ2 JavaScript Pivotview control

Member Sorting

Allows to order field members in rows and columns either in ascending or descending order. By default, field members in rows and columns are in ascending order.

Drill through in EJ2 JavaScript Pivotview control

Editing in EJ2 JavaScript Pivotview control

This feature is applicable only for the relational data source.

Data Formatting

Number formatting in EJ2 JavaScript Pivotview control

Allows you to specify the required display format that should be used in values of the pivot table.

Supported display formats are:

- Number
- Currency
- Percentage
- Custom

Conditional formatting in EJ2 JavaScript Pivotview control

Allows end user to change the appearance of the pivot table value cells with its background color, font color, font family, and font size based on specific conditions.

Report Manipulation

Grouping bar in EJ2 JavaScript Pivotview control

Field list in EJ2 JavaScript Pivotview control

The pivot table provides a built-in Field List similar to Microsoft Excel. It allows to add or remove fields and also rearrange them between different axes, including column, row, value, and filter along with sort and filter options dynamically at runtime.

The field list can be displayed in two different formats to interact with pivot table. They are:

- **In-built Field List (Popup):** To display the field list icon in pivot table UI to invoke the built-in dialog.
- **Stand-alone Field List (Fixed):** To display the field list in a static position within a web page.

In-built Field List (Popup)

To enable the field list in pivot table UI, set the [showFieldList](#) property in pivot table to **true**. A small icon will appear on the top left corner of the pivot table and clicking on this icon, field list dialog will appear.

The field list icon will be displayed at the top right corner of the pivot table, when grouping bar is enabled.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    allowLabelFilter: true,
    allowValueFilter: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
},
```

```

        height: 350,
        showFieldList: true
    });
    pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stand-alone Field List (Fixed)

The field list can be rendered in a static position, anywhere in web page layout, like a separate component. To do so, you need to set [renderMode](#) property to **Fixed** in [pivotFieldList](#).

To make field list interact with pivot table, you need to use the [updateView](#) and [update](#) methods for data source update in both field list and pivot table simultaneously.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    height: 300,
    enginePopulated: function () {
        if (fieldlistObj) {
            fieldlistObj.update(pivotTableObj);
        }
    }
});
pivotTableObj.appendTo('#PivotTable');
var fieldlistObj = new ej.pivotview.PivotFieldList({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    renderMode: 'Fixed',
    enginePopulated: function () {
        fieldlistObj.updateView(pivotTableObj);
    }
});
fieldlistObj.appendTo('#Static_FieldList');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="PivotTable"></div>
        <br>
        <div id="Static_FieldList"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Search desired field

End user can search for desired field in the field list UI by typing the field name into the search box at runtime. It can be enabled by setting the [enableFieldSearching](#) property to **true** via code-behind.

By default, field search option is disabled in the field list UI.

To enable search box in the static field list UI, set the [enableFieldSearching](#) property to **true** in [PivotFieldList](#).

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    enginePopulated: function () {
        if (fieldlistObj) {

```



```

        fieldlistObj.update(pivotTableObj);
    }
}
});
pivotTableObj.appendTo('#PivotTable');
var fieldlistObj = new ej.pivotview.PivotFieldList({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    renderMode: 'Fixed',
    enginePopulated: function () {
        fieldlistObj.updateView(pivotTableObj);
    },
    enableFieldSearching: true
});
fieldlistObj.appendTo('#Static FieldList');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="PivotTable"></div>
        <br>
        <div id="Static_FieldList"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To enable search box in the pivot table's built-in popup field list UI, set the [enableFieldSearching](#) property to **true** in [PivotView](#).

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Products' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    showFieldList: true,
    enableFieldSearching: true,
    enginePopulated: function (args) {
        Object.keys(pivotTableObj.engineModule.fieldList).forEach((key,
index) => {
            if (key === 'Quarter') {
                pivotTableObj.engineModule.fieldList[key].caption =
'Production Quarter Year';
            }
            else if (key === 'Year') {
                pivotTableObj.engineModule.fieldList[key].caption =
'Production Year';
            }
        });
    }
});

```

```
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

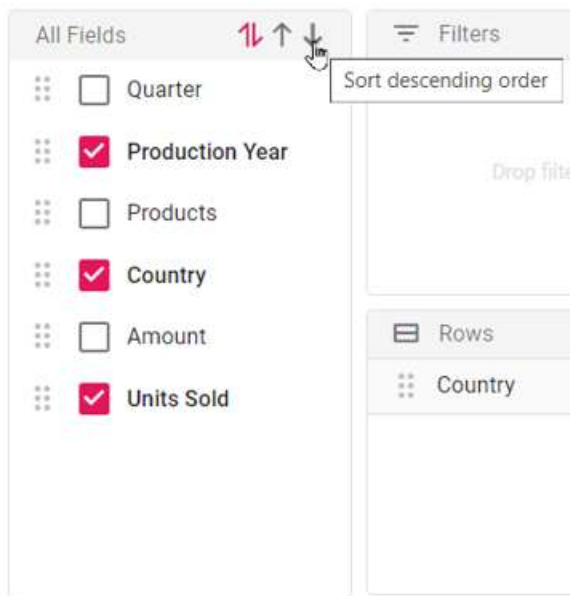
```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Option to sort fields

End user can sort fields in the field list UI to ascending (or) descending (or) default order (as obtained from the data source) using the built-in sort icons.

By default, fields are displayed in the default order.

Field List



Sort fields in a desired order

To display the fields in descending order by default, set the [defaultFieldListOrder](#) property to **Descending** in the [load](#) event.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    allowLabelFilter: true,
    allowValueFilter: true,
    columns: [{ name: 'Year', caption: 'Production Year' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }],
    rows: [{ name: 'Country' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: [],
  },
  showGroupingBar: true,
  showFieldList: true,
  height: 350,
```

```

    load: function(args) {
        args.defaultFieldListOrder = 'Descending';
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Group fields under desired folder name

In the field list UI, you can display fields by grouping them under the desired folder name. It can only be configured via code-behind by setting the [groupName](#) property in [fieldMapping](#).

You can only group fields to one level using the [groupName](#) property.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        allowLabelFilter: true,
        allowValueFilter: true,
        columns: [{ name: 'Year', caption: 'Production Year' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }],
        rows: [{ name: 'Country' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        fieldMapping: [
            { name: 'Quarter', groupName: 'Product category' },
            { name: 'Products', groupName: 'Product category' },
            { name: 'Amount', groupName: 'Product category', caption: 'Sold
Amount' },
        ]
    },
    height: 350,
    showFieldList: true,
    showGroupingBar: true
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

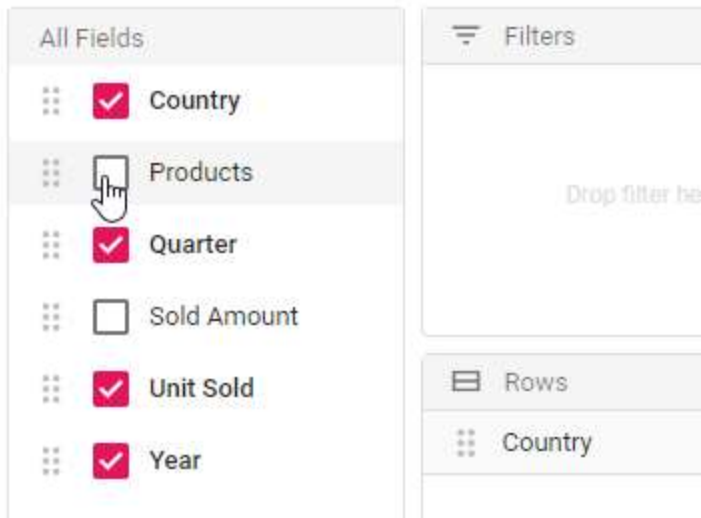
    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add or remove fields

Using check box besides each field, end user can select or unselect to add or remove fields respectively from the report at runtime.

Field List



Remove specific field(s) from displaying

When a data source is bound to the component, fields will be automatically populated inside the Field List. In such case, user can also restrict specific field(s) from displaying. To do so, set the appropriate field name(s) in [excludeFields](#) property belonging to [dataSourceSettings](#).

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    allowLabelFilter: true,
    allowValueFilter: true,
    columns: [{ name: 'Year', caption: 'Production Year' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: [],
    excludeFields: ['Quarter']
  },
  height: 350,
  showFieldList: true
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
```



```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Re-arranging fields

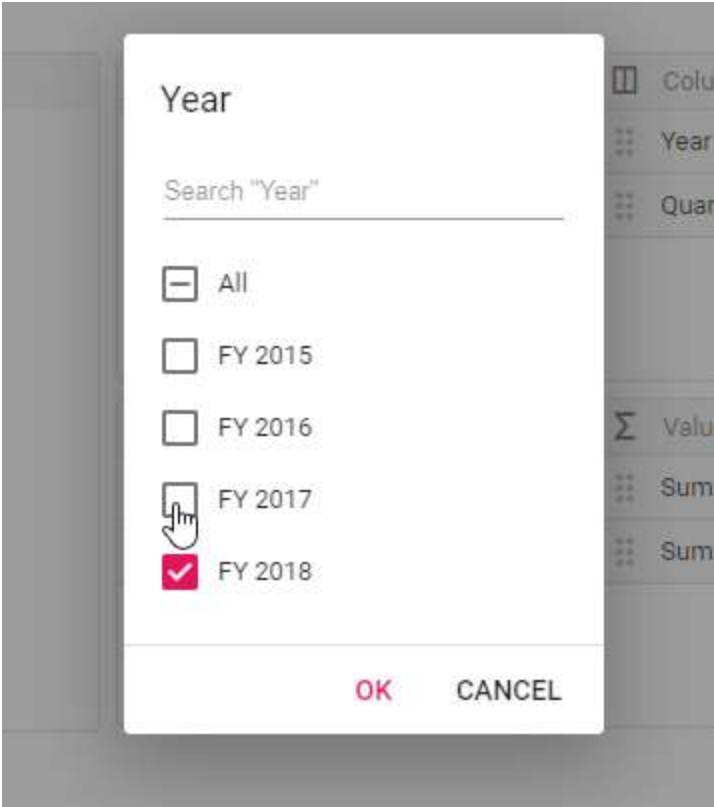
In-order to re-arrange, drag any field from the field list and drop it into the column, row, value, or filter axis using the drag-and-drop holder. It helps end user to alter the report at runtime.

<div>Filters</div> <div>Products (All)</div>	<div>Columns</div> <div>Year</div> <div>Quarter</div>
<div>Rows</div> <div>Country</div>	<div>Values</div> <div>Sum of Unit Sold</div> <div>Sum of Sold Amount</div>

Filtering members

Using the filter icon besides each field in row, column and filter axes, members can be either included or excluded at runtime. To know more about member filtering, [refer](#) here.

<div>Filters</div> <div>Drop filter here</div>	<div>Columns</div> <div>Year</div> <div>Quarter</div>
<div>Rows</div> <div>Country</div> <div>Products</div>	<div>Values</div> <div>Sum of Unit Sold</div> <div>Sum of Sold Amount</div>



	FY 2018		Grand Total	
	Units Sold	Sold Amount	Units Sold	Sold Amount
▶ France	16	\$27,264	16	\$27,264
▶ Germany	96	\$77,264	96	\$77,264
▶ United States	76	\$97,264	76	\$97,264
Grand Total	188	\$201,792	188	\$201,792

Sorting members

Using the sort icon besides each field in row and column axes, members can be arranged either in ascending or descending order at runtime. To know more about member sorting, [refer](#) here.

≡ Filters

Drop filter here

☐ Columns

⋮ Year

↑ ≡

⋮ Quarter

↑ ≡

☰ Rows

⋮ Country

⋮ Products

Σ Values

⋮ Sum of Unit Sold

⋮ Sum of Sold Amount

	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
United States	546	\$754,515	636	\$2,263,104	
Germany	440	\$563,515	496	\$1,772,104	
France	450	\$714,955	526	\$1,542,104	
Grand Total	1436	\$2,032,985	1658	\$5,577,312	1

Calculated fields

The calculated field support allows end user to add a new calculated field based on the available fields from the bound data source using basic arithmetic operators. To enable this support in Field List UI, set the `allowCalculatedField` property to `true` in pivot table. Now a button will be seen automatically inside the field list UI which will invoke the calculated field dialog on click. To know more about calculated field, [refer](#) here.

Field List

ALL FIELDS

Country

Products

Quarter

Sold Amount

Total Amount

Unit Sold

Year

Filters

Drop filter here

Rows

Country

Products

CALCULATED FIELD

Columns

Year

Quarter

Σ Values

Sum of Unit Sold

Sum of Sold Amount

Total Amount

CLOSE

Copyright © 2001 -2024 Syncfusion Inc.

2809

Create Calculated Field

×

Total Amount

Drag and drop fields to formula

Total Amount (CalculatedField)

✎

Unit Sold (Sum)

>

Year (Count)

Formula

"Sum(Amount)*"Sum(Sold)"

OK

CANCEL

	FY 2015			FY 2016	
	Units Sold	Sold Amount	Total Amount	Units Sold	Sold Amount
France	450	\$714,955	\$715,405	526	\$1,542,
Germany	440	\$563,515	\$563,955	496	\$1,772,
United States	546	\$754,515	\$755,061	636	\$2,263,
Grand Total	1436	\$2,032,985	\$2,034,421	1658	\$5,577,

Changing aggregation type of value fields at runtime

End user can perform calculations over a group of values using the aggregation option. The value fields bound to the field list, appears with a dropdown icon, helps to select an appropriate aggregation type at runtime. On selection, the values in the Pivot Table will be changed dynamically. To know more about aggregation, [refer](#) here.

Filters

Drop filter here

Columns

Year

Quarter

Rows

Country

Products

Values

Sum of Unit Sold

Sum of Sold Amount

Rows

Country

Products

Values

Sum of Unit Sold

Sum of Sold Amount

Sum

Count

Distinct Count

Product

Avg

Min

Max

More...

	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
United States	68.25	\$754,515	79.5	\$2,263,104	91.
Germany	55	\$563,515	62	\$1,772,104	
France	56.25	\$714,955	65.75	\$1,542,104	
Grand Total	59.83333333...	\$2,032,985	69.08333333...	\$5,577,312	70.

Defer layout update

Defer layout update support to update the pivot table only on demand and not during every user action. To enable this support in Field List UI, set the [allowDeferLayoutUpdate](#) property to **true** in pivot table. Now a check box inside Field List UI will be seen in checked state, allowing pivot table to update only on demand. To know more about defer layout, [refer](#) here.

☒ Unit Sold
☒ Year

Rows

Country ↑

Products ↑

Σ Values

Sum of Unit Sold ▼

Sum of Sold Amount ▼

☒ Defer Layout Update

APPLY
 CANCEL

Show built-in Field List (Popup) over specific target

By passing the target element to the built-in field list dialog module in the [dataBound](#) event, the field list dialog will be displayed over the appropriate target element on a web page. By default, the Pivot Table's parent element is used as the target element to display the built-in field list dialog.

The sample code below shows the built-in field list dialog using `document.body` as the target element.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    allowLabelFilter: true,
    allowValueFilter: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
```



```

        filters: []
    },
    height: 350,
    dataBound: function (args) {
        // Here the target can be set to the built-in field list dialog

pivotTableObj.pivotFieldListModule.dialogRenderer.fieldListDialog.target =
document.body;
    },
    showFieldList: true
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div>
    <div id="PivotTable"></div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show field list using toolbar

It can also be viewed in toolbar by setting [showFieldList](#) and [showToolbar](#) properties in pivot table to **true**. Also, include the item **FieldList** within the [toolbar](#) property in pivot table. When toolbar is enabled, field list icon will be automatically added into the toolbar and the icon won't appear on top left corner in the pivot table component.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  showToolbar: true,
  toolbar: ['FieldList'],
  showFieldList: true,
  height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Invoking dynamic Field List (Customized)

Also, you can display the field list dialog independently through other means. For example, you can invoke the field list dialog on an external button click. To do so, set **renderMode** property to **Popup** and since on button click, field list dialog will be invoked.

* Meanwhile, you can display the field list dialog over specific target element within a webpage using **target** property. By default, the **target** value is "null", which by default refers the **document.body** element.

* To make field list interact with pivot table, you need to use the **updateView** and **update** methods for data source update in both field list and pivot table simultaneously.

The below sample code illustrates the field list dialog invoked on an external button click.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
    height: 350,
    enginePopulated: function () {
        if (fieldlistObj) {
            fieldlistObj.update(pivotTableObj);
        }
    }
});
pivotTableObj.appendTo('#PivotTable');
var fieldlistObj = new ej.pivotview.PivotFieldList({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    renderMode: 'Popup',
    target: '#PivotFieldList',
    enginePopulated: function () {
        fieldlistObj.updateView(pivotTableObj);
    }
});
fieldlistObj.appendTo('#PivotFieldList');
var fieldListBtn = new ej.buttons.Button({ isPrimary: true });
fieldListBtn.appendTo('#fieldlistbtn');
document.getElementById('fieldlistbtn').onclick = function () {
    fieldlistObj.dialogRenderer.fieldListDialog.show();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="fieldlistbtn" type="button" value="Field List"
name="Field List" title="Field List">
        <div id="PivotFieldList"></div>
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Set caption to fields which isn't bound to the report

<!-- markdownlint-disable MD009 -->

One can set the caption to all fields from the data source even if it is not bound to the actual report. It can be achieved using the [enginePopulated](#) event. On doing so, caption of the respective field will be displayed in both grouping bar and field list.

In the sample, we have set caption to the fields **Year** and **Quarter** dynamically.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Products' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }],
    },

```

```

        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    showFieldList: true,
    enginePopulated: function (args) {
        Object.keys(pivotTableObj.engineModule.fieldList).forEach((key,
index) => {
            if (key === 'Quarter') {
                pivotTableObj.engineModule.fieldList[key].caption =
'Production Quarter Year';
            }
            else if (key === 'Year') {
                pivotTableObj.engineModule.fieldList[key].caption =
'Production Year';
            }
        });
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show values button

During runtime, the **Values** button in the field list can be moved to a different position (i.e., different index) among other fields in the column or row axis. To enable the **Values** button, set the [showValuesButton](#) property to **true**.

This support is only available for relational data sources.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    height: 300,
    enginePopulated: function () {
        if (fieldlistObj) {
            fieldlistObj.update(pivotTableObj);
        }
    }
});
pivotTableObj.appendTo('#PivotTable');
var fieldlistObj = new ej.pivotview.PivotFieldList({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    renderMode: 'Fixed',
    showValuesButton: true,
    enginePopulated: function () {

```

```

        fieldlistObj.updateView(pivotTableObj);
    }
});
fieldlistObj.appendTo('#Static_FieldList');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="PivotTable"></div>
    <br>
    <div id="Static_FieldList"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>

```



```
</body></html>
```

Events

EnginePopulated

The [enginePopulated](#) event is available in both Pivot Table and Field List.

- The event [enginePopulated](#) is triggered in field list whenever the report gets modified. The updated report is passed to the pivot table via [updateView](#) method written within this event to refresh the same.
- Likewise, [enginePopulated](#) event is triggered in pivot table whenever the report gets modified. The updated report is passed to the field list via [update](#) method written within this event to refresh the same.

The event [enginePopulated](#) is triggered after engine is populated. It has following parameters - [dataSourceSettings](#), [pivotFieldList](#) and [pivotValues](#).

Note: This event is not required for Popup field list since it is a in built one.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  height: 300,
  enginePopulated: function () {
    if (fieldlistObj) {
      fieldlistObj.update(pivotTableObj);
    }
  }
});
pivotTableObj.appendTo('#PivotTable');
var fieldlistObj = new ej.pivotview.PivotFieldList({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  renderMode: 'Fixed',
  enginePopulated: function () {
    fieldlistObj.updateView(pivotTableObj);
  }
});
fieldlistObj.appendTo('#Static_FieldList');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="PivotTable"></div>
        <br>
        <div id="Static_FieldList"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

FieldListRefreshed

The event [FieldListRefreshed](#) is triggered whenever there is any change done in the field list UI. It has following parameter - **dataSourceSettings** and **pivotValues**. It allows user to identify each field list update. This event is applicable only for static field list.

For example, if we perform a sort operation within the field list, the field list will be refreshed. The [fieldListRefreshed](#) event will be triggered at that time and the user can perform custom operation inside that event.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    allowLabelFilter: true,
    allowValueFilter: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  fieldListRefreshed: function (args) {
    //Triggers, whenever field list get refreshed.
  },
  height: 350,
  showFieldList: true
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

FieldDropped

The event [onFieldDropped](#) fires whenever a field is dropped in an axis. It has following parameters - **droppedAxis**, **DroppedField** and **DataSourceSettings**. In this illustration, we have modified the **DroppedField** caption through this event at runtime.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        allowLabelFilter: true,
        allowValueFilter: true,
        columns: [{ name: 'Year', caption: 'Production Year' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    showFieldList: true,
    onFieldDropped: function (args) {

```

```

        args.droppedField.caption = args.droppedField.name + " --> " +
args.droppedAxis;
    },
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ActionBegin

The event [actionBegin](#) triggers when the UI actions such as sorting, filtering, aggregation or edit calculated field, that are present in the field list UI begin. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action began. The following are the UI actions and their names:

Action	Action Name
-----	-----
Sort icon	Sort field
Filter icon	Filter field
Aggregation (Value type drop down and menu)	Aggregate field
Edit icon	Edit calculated field
Calculated field button	Open calculated field dialog
Field list	Open field list
Field list tree – Sort icon	Sort field tree

- **fieldInfo**: It holds the selected field information.

Note: This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **cancel**: It allows user to restrict the current action.

In the below sample, opening pop-up field list can be restricted by setting the **args.cancel** option to **true** in the **actionBegin** event.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        allowLabelFilter: true,
        allowValueFilter: true,

```

```

        columns: [{ name: 'Year', caption: 'Production Year' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    showGroupingBar: true,
    showFieldList: true,
    actionBegin: function (args) {
        if (args.actionName == 'Open field list') {
            args.cancel = true;
        }
    },
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div class="container">
      <div id="PivotTable"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ActionComplete

The event [actionComplete](#) triggers when the UI actions such as sorting, filtering, aggregation or edit calculated field, that are present in the field list UI, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. The following are the UI actions and their names:

Action	Action Name
-----	-----
Sort icon	Field sorted
Filter icon	Field filtered
Aggregation (Value type drop down and menu)	Field aggregated
Edit icon	Calculated field edited
Calculated field button	Calculated field applied
Field list	Field list closed
Field list tree – Sort icon	Field tree sorted

- **fieldInfo**: It holds the selected field information.

Note: This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **actionInfo**: It holds the unique information about the current UI action. For example, if sorting is completed, the event argument contains information such as sort order and the field name.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,

```



```

        expandAll: false,
        enableSorting: true,
        allowLabelFilter: true,
        allowValueFilter: true,
        columns: [{ name: 'Year', caption: 'Production Year' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    showGroupingBar: true,
    showFieldList: true,
    actionComplete: function (args) {
        if (args.actionName == 'Field list closed') {
            // Triggers when the field list dialog is closed.
        }
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ActionFailure

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- **actionName**: It holds the name of the current action failed. The following are the UI actions and their names:

Action	Action Name
-----	-----
Sort icon	Sort field
Filter icon	Filter field
Aggregation (Value type drop down and menu)	Aggregate field
Edit icon	Edit calculated field
Calculated field button	Open calculated field dialog
Field list	Open field list
Field list tree – Sort icon	Sort field tree

- **errorInfo**: It holds the error information of the current UI action.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],

```

```

        filters: []
    },
    showGroupingBar: true,
    showFieldList: true,
    allowCalculatedField: true,
    showToolbar: true,
    displayOption: { view: 'Both' },
    toolbar: ['New', 'Save', 'Rename', 'Remove', 'Load',
        'Grid', 'Chart', 'MDX', 'Export', 'SubTotal', 'GrandTotal',
        'ConditionalFormatting', 'FieldList'],
    allowExcelExport: true,
    allowConditionalFormatting: true,
    allowPdfExport: true,
    actionFailure: function (args) {
        if (args.actionName == 'Open field list') {
            // Triggers when the current UI action fails to achieve the
            desired result.
        }
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Customize the icons for pivot table](#)

Defer update in EJ2 JavaScript Pivotview control

Defer layout update support allows to update the pivot table component only on demand. On enabling this feature, end user can drag-and-drop fields between row, column, value and filter axes, apply sorting and filtering inside the Field List, resulting in change of pivot report alone but not the pivot table values. Once all operations are performed and on clicking the "Apply" button in the Field List, pivot table will start to update with the last modified report. This also helps in bringing better performance in pivot table component rendering.

The field list can be displayed in two different formats to interact with pivot table. They are:

- **In-built Field List (Popup):** To display the field list icon in pivot table UI to invoke the built-in dialog.
- **Stand-alone Field List (Fixed):** To display the field list in a static position within a web page.

In-built Field List (Popup)

To enable deferred updates in the pivot table, set the property [allowDeferLayoutUpdate](#) in pivot table as **true**. To make a note, the defer update option can be controlled only via Field List during runtime.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],

```

```

        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    allowDeferLayoutUpdate: true,
    showFieldList: true,
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stand-alone Field List (Fixed)

The field list can be rendered in a static position, anywhere in web page layout, like a separate component. To do so, you need to set [renderMode](#) property to **Fixed** in [pivotFieldList](#).

To enable deferred updates in the static fieldlist, set the property [allowDeferLayoutUpdate](#) in [pivotFieldlist](#) as **true**. To make a note, the defer update option can be controlled only via Field List during runtime.

To make field list interact with pivot table, you need to use the **updateView** and **update** methods for data source update in both field list and pivot table simultaneously.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    allowDeferLayoutUpdate: true,
    enginePopulated: function () {
        if (fieldlistObj) {
            fieldlistObj.update(pivotTableObj);
        }
    },
    height: 350,
});
pivotTableObj.appendTo('#PivotTable');
var fieldlistObj = new ej.pivotview.PivotFieldList({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    allowDeferLayoutUpdate: true,
    renderMode: 'Fixed',
    enginePopulated: function () {
        if (fieldlistObj.isRequiredUpdate) {
            fieldlistObj.updateView(pivotTableObj);
        }
        pivotTableObj.notify('ui-update', pivotTableObj);
    }
});

```

```

        fieldlistObj.notify('tree-view-update', fieldlistObj);
    }
});
fieldlistObj.appendTo('#Static_FieldList');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="PivotTable"></div>
    <br>
    <div id="Static_FieldList"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Performance

```
<!-- markdownlint-disable MD036 -->
```

Virtual scrolling in EJ2 JavaScript Pivotview control

Virtual scrolling

Allows to load the large amounts of data without any performance degradation by rendering rows and columns only in the current content viewport. Rest of the aggregated data will be brought into viewport dynamically based on vertical or horizontal scroll position. This feature can be enabled by setting the [enableVirtualization](#) property in pivot table to **true**..

To use the virtual scrolling feature, inject the **VirtualScroll** module in to the pivot table.

INDEX.JS

```
var customername = ['TOM', 'Hawk', 'Jon', 'Chandler', 'Monica',
  'Rachel', 'Phoebe', 'Gunther',
    'Ross', 'Geller', 'Joey', 'Bing', 'Tribbiani', 'Janice', 'Bong',
  'Perk', 'Green', 'Ken', 'Adams'];
var city = ['New York', 'Los Angeles', 'Chicago', 'Houston',
  'Philadelphia', 'Phoenix', 'San Antonio', 'Austin',
    'San Francisco', 'Columbus', 'Washington', 'Portland', 'Oklahoma',
  'Las Vegas', 'Virginia', 'St. Louis', 'Birmingham'];
var data = function (count) {
  var result = [];
  var dt = 0;
  for (var i = 1; i < (count + 1); i++) {
    dt++;
    var round = void 0;
    var toString_1 = i.toString();
    if (toString_1.length === 1) {
      round = '0000' + (i);
    }
    else if (toString_1.length === 2) {
      round = '000' + i;
    }
    else if (toString_1.length === 3) {
      round = '00' + i;
    }
    else if (toString_1.length === 4) {
      round = '0' + i;
    }
    else {
      round = toString_1;
    }
    result.push({
      ProductID: 'PRO-' + round,
      City: city[Math.round(Math.random() * city.length)] ||
city[0],
      Year: "FY " + (dt + 2013),
      CustomerName: customername[Math.round(Math.random() *
customername.length)] || customername[0],
      Price: Math.round(Math.random() * 5000) + 5000,
      Sold: Math.round(Math.random() * 80) + 10,
```



```

    });
    if (dt / 4 == 1) {
        dt = 0;
    }
}
return result;
};
var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: data(1000),
        enableSorting: false,
        expandAll: true,
        formatSettings: [{ name: 'Price', format: 'C0' }],
        rows: [{ name: 'ProductID' }],
        columns: [{ name: 'Year' }],
        values: [{ name: 'Price', caption: 'Unit Price' }, { name:
'Sold', caption: 'Unit Sold' }]
    },
    height: 350,
    enableVirtualization: true
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

```

```

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations for virtual scrolling

- In virtual scrolling, the `columnWidth` property in `gridSettings` should be in pixel and percentage values are not accepted.
- Resizing columns, setting width to individual columns which affects the calculation used to pick the correct page on scrolling.
- Grouping, which takes additional time to splitting the raw items into the provided format.
- Date Formatting, which takes additional time to convert date format.
- Date Formatting with sorting, here additionally full date time format should be framed to perform sorting along with the provided date format which lags the performance.
- When using OLAP data, subtotals and grandtotals are only displayed when measures are bound at the last position in the [rows](#) or [columns](#) axis. Otherwise, the data from the pivot table will be shown without summary totals.
- Even if virtual scrolling is enabled, not only is the current view port data retrieved, but also the data for the immediate previous page and the immediate next page. As a result, when the end user scrolls slightly ahead or behind, the next or previous page data is displayed immediately without requiring a refresh. **Note:** If the pivot table's width and height are large, the loading data count in the current, previous, and next view ports (pages) will also increase, affecting performance.

Data Compression

This property is applicable only for relational data source.

When we bind one million raw data, the pivot table will process all raw data to generate aggregated data during initial rendering and report manipulation. But in data compression, the data will be compressed based on the uniqueness of the raw data, and unique records will be provided as input for

the Pivot Table. The compressed data will be used for further operations at all times, reducing the looping complexity and improving the performance of the pivot table. For example, if the pivot table is connected to one million raw data aggregated to 1,000 unique data means, it will be rendered within 3 seconds rather than 10 seconds. You can enable this option by using the [allowDataCompression](#) property along with [enableVirtualization](#) property.

This options will only function when the virtual scrolling is enabled.

INDEX.JS

```

var customername = ['TOM', 'Hawk', 'Jon', 'Chandler', 'Monica',
'Rachel', 'Phoebe', 'Gunther',
    'Ross', 'Geller', 'Joey', 'Bing', 'Tribbiani', 'Janice', 'Bong',
'Perk', 'Green', 'Ken', 'Adams'];
var city = ['New York', 'Los Angeles', 'Chicago', 'Houston',
'Philadelphia', 'Phoenix', 'San Antonio', 'Austin',
    'San Francisco', 'Columbus', 'Washington', 'Portland', 'Oklahoma',
'Las Vegas', 'Virginia', 'St. Louis', 'Birmingham'];
var data = function (count) {
    var result = [];
    var dt = 0;
    for (var i = 1; i < (count + 1); i++) {
        dt++;
        var round = void 0;
        var toString_1 = i.toString();
        if (toString_1.length === 1) {
            round = '0000' + (i);
        }
        else if (toString_1.length === 2) {
            round = '000' + i;
        }
        else if (toString_1.length === 3) {
            round = '00' + i;
        }
        else if (toString_1.length === 4) {
            round = '0' + i;
        }
        else {
            round = toString_1;
        }
        result.push({
            ProductID: 'PRO-' + round,
            City: city[Math.round(Math.random() * city.length)] ||
city[0],
            Year: "FY " + (dt + 2013),
            CustomerName: customername[Math.round(Math.random() *
customername.length)] || customername[0],
            Price: Math.round(Math.random() * 5000) + 5000,
            Sold: Math.round(Math.random() * 80) + 10,
        });
        if (dt / 4 == 1) {
            dt = 0;
        }
    }
    return result;
};
var pivotTableObj = new ej.pivotview.PivotView({

```

```

        dataSourceSettings: {
            dataSource: data(1000),
            enableSorting: false,
            expandAll: true,
            formatSettings: [{ name: 'Price', format: 'C0' }],
            rows: [{ name: 'ProductID' }],
            columns: [{ name: 'Year' }],
            values: [{ name: 'Price', caption: 'Unit Price' }, { name:
'Sold', caption: 'Unit Sold' }]
        },
        height: 350,
        enableVirtualization: true
    });
    pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations during data compression

- The following aggregation types will not be supported.
- Average
- Populationstdev
- Samplestdev
- Populationvar
- Samplevar
- If you use any of the aggregations above, it will result in an aggregation type **"Sum"**.
- Distinctcount will act as **"Count"** aggregation type.
- In the calculated field, an existing field can be inserted without altering its default aggregation type. Even if we change it, it would use the default aggregation type back for calculation.

Virtual scrolling for static field list

Virtual scrolling automatically works with "Popup" field list on setting the [enableVirtualization](#) property in the Pivot Table to **true**. In case of static field list, which act as a separate component, user need to enable [enableVirtualization](#) property in the Pivot Table and also pass the report information to pivot table instance via the [load](#) event of the field list.

INDEX.JS

```

var data = function (count) {
  var result = [];
  var dt = 0;
  for (var i = 1; i < (count + 1); i++) {
    dt++;
    var round = void 0;
    var toString_1 = i.toString();
    if (toString_1.length === 1) {
      round = '0000' + (i);
    }
    else if (toString_1.length === 2) {
      round = '000' + i;
    }
    else if (toString_1.length === 3) {
      round = '00' + i;
    }
  }
}

```

```

        else if (toString_1.length === 4) {
            round = '0' + i;
        }
        else {
            round = toString_1;
        }
        result.push({
            ProductID: 'PRO-' + round,
            Year: "FY " + (dt + 2013),
            Price: Math.round(Math.random() * 5000) + 5000,
            Sold: Math.round(Math.random() * 80) + 10,
        });
        if (dt / 4 == 1) {
            dt = 0;
        }
    }
    return result;
};
var pivotObj = new ej.pivotview.PivotView({
    height: 300,
    enableVirtualization: true,
    enginePopulated: function () {
        if (fieldListObj) {
            fieldListObj.update(pivotObj);
        }
    },
});
pivotObj.appendTo('#PivotTable');
var fieldListObj = new ej.pivotview.PivotFieldList({
    dataSourceSettings: {
        dataSource: data(1000),
        rows: [{ name: 'ProductID' }],
        columns: [{ name: 'Year' }],
        values: [{ name: 'Price', caption: 'Unit Price' }, { name:
'Sold', caption: 'Unit Sold' }]
    },
    renderMode: 'Fixed',
    enginePopulated: function () {
        fieldListObj.updateView(pivotObj);
    },
    load: function () {
        this.pivotGridModule = pivotObj;
        //Assigning report to pivot table component.
        pivotObj.dataSourceSettings = fieldListObj.dataSourceSettings;
        //Generating page settings based on pivot table component's
size.
        pivotObj.updatePageSettings(true);
        //Assigning page settings to field list component.
        fieldListObj.pageSettings = pivotObj.pageSettings;
    },
});
fieldListObj.appendTo('#Static_FieldList');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Pivot Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="PivotTable"></div>
        <br>
        <div id="Static_FieldList"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can also explore our [Virtual Scrolling in JavaScript PivotTable](#) that showing a large number of records.

Paging in EJ2 JavaScript Pivotview control

Paging allows you to load large amounts of data that can be divided and displayed page by page in the pivot table. It can be enabled by setting the [enablePaging](#) property to **true**. It can be configured at code-behind by using the [pageSettings](#) property, during initial rendering of the component. The properties required are:

- [currentRowPage](#): Allows user to set the current row page number to be displayed in the pivot table.
- [currentColumnPage](#): Allows user to set the current column page number to be displayed in the pivot table.
- [rowPageSize](#): Allows user to set the total number of records to be displayed on each page of the pivot table's row axis.
- [columnPageSize](#): Allows user to set the total number of records to be displayed on each page of the pivot table's column axis.

Pager UI

When paging is enabled, a built-in pager UI appears at the bottom of the pivot table, allowing you to change the current page in the row and column axes by navigating to a desired page using the navigation buttons or an input text box, as well as change the page size via dropdown at runtime.

You can also change the position, visibility, compact view, and template of the row and column pagers by using the [pagerSettings](#).

In order to see and use the pager UI, insert the **Pager** module into the pivot table.

INDEX.JS

```
var remoteData = new ej.data.DataManager({
  url: 'https://bi.syncfusion.com/northwindservice/api/orders',
  adaptor: new ej.data.WebApiAdaptor(),
  crossDomain: true
});
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    type: 'JSON',
    dataSource: remoteData,
    expandAll: true,
    columns: [{
      name: 'ProductName',
      caption: 'Product Name'
    }],
    rows: [{
      name: 'ShipCountry',
      caption: 'Ship Country'
    }, {
      name: 'ShipCity',
      caption: 'Ship City'
    }],
    formatSettings: [{
      name: 'UnitPrice',
      format: 'C0'
    }],
    values: [{
      name: 'Quantity'
```



```

    }, {
        name: 'UnitPrice',
        caption: 'Unit Price'
    }],
    filters: []
},
width: '100%',
height: 350,
enablePaging: true,
pageSettings: {
    rowPageSize: 10,
    columnPageSize: 5,
    currentColumnPage: 1,
    currentRowPage: 1
},
pagerSettings: {
    position: 'Bottom',
    enableCompactView: false,
    showColumnPager: true,
    showRowPager: true,
    columnPageSizes: [5, 10, 20, 50, 100],
    rowPageSizes: [10, 50, 100, 200],
    isInversed: false,
    showColumnPageSize: true,
    showRowPageSize: true
},
gridSettings: {
    columnWidth: 120
},
});
pivotObj.appendTo("#PivotTable");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show pager UI at top or bottom

You can display the pager UI at top or bottom of the pivot table by using the [position](#) property. To show the pager UI at top of the pivot table, set the [position](#) property in [pagerSettings](#) to **Top**.

By default, the pager UI appears at the bottom of the pivot table.

INDEX.JS

```

var remoteData = new ej.data.DataManager({
    url: 'https://bi.syncfusion.com/northwindservice/api/orders',
    adaptor: new ej.data.WebApiAdaptor(),
    crossDomain: true
});
var pivotObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        type: 'JSON',
        dataSource: remoteData,
        expandAll: true,
        columns: [{
            name: 'ProductName',
            caption: 'Product Name'
        }],
    },

```

```

        rows: [{
            name: 'ShipCountry',
            caption: 'Ship Country'
        }, {
            name: 'ShipCity',
            caption: 'Ship City'
        }],
        formatSettings: [{
            name: 'UnitPrice',
            format: 'C0'
        }],
        values: [{
            name: 'Quantity'
        }, {
            name: 'UnitPrice',
            caption: 'Unit Price'
        }],
        filters: []
    },
    width: '100%',
    height: 350,
    enablePaging: true,
    pageSettings: {
        rowPageSize: 10,
        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
    },
    pagerSettings: {
        position: 'Top'
    },
    gridSettings: {
        columnWidth: 120
    },
    },
    });
pivotObj.appendTo("#PivotTable");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Inverse pager

Toggles and displays row and column pager. To show the column pager on the left side of the pager UI, set the [isInversed](#) property in [pagerSettings](#) to **true**.

By default, the row pager is displayed on the left side of the pager UI, while the column pager is displayed on the right side.

INDEX.JS

```

var remoteData = new ej.data.DataManager({
    url: 'https://bi.syncfusion.com/northwindservice/api/orders',
    adaptor: new ej.data.WebApiAdaptor(),
    crossDomain: true
});
var pivotObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        type: 'JSON',
        dataSource: remoteData,

```

```

        expandAll: true,
        columns: [{
            name: 'ProductName',
            caption: 'Product Name'
        }],
        rows: [{
            name: 'ShipCountry',
            caption: 'Ship Country'
        }, {
            name: 'ShipCity',
            caption: 'Ship City'
        }],
        formatSettings: [{
            name: 'UnitPrice',
            format: 'C0'
        }],
        values: [{
            name: 'Quantity'
        }, {
            name: 'UnitPrice',
            caption: 'Unit Price'
        }],
        filters: []
    },
    width: '100%',
    height: 350,
    enablePaging: true,
    pageSettings: {
        rowPageSize: 10,
        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
    },
    pagerSettings: {
        isInversed: true
    },
    gridSettings: {
        columnWidth: 120
    },
    });
pivotObj.appendTo("#PivotTable");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Compact view

By hiding all except the previous and next navigation buttons, the pager UI can be displayed with the absolute minimum of paging options. The compact view can be enabled by setting the [enableCompactView](#) property in [pagerSettings](#) to **true**.

INDEX.JS

```

var remoteData = new ej.data.DataManager({
    url: 'https://bi.syncfusion.com/northwindservice/api/orders',
    adaptor: new ej.data.WebApiAdaptor(),
    crossDomain: true
});

```

```

var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    type: 'JSON',
    dataSource: remoteData,
    expandAll: true,
    columns: [{
      name: 'ProductName',
      caption: 'Product Name'
    }],
    rows: [{
      name: 'ShipCountry',
      caption: 'Ship Country'
    }, {
      name: 'ShipCity',
      caption: 'Ship City'
    }],
    formatSettings: [{
      name: 'UnitPrice',
      format: 'C0'
    }],
    values: [{
      name: 'Quantity'
    }, {
      name: 'UnitPrice',
      caption: 'Unit Price'
    }],
    filters: []
  },
  width: '100%',
  height: 350,
  enablePaging: true,
  pageSettings: {
    rowPageSize: 10,
    columnPageSize: 5,
    currentColumnPage: 1,
    currentRowPage: 1
  },
  pagerSettings: {
    enableCompactView: true
  },
  gridSettings: {
    columnWidth: 120
  },
});
pivotObj.appendTo("#PivotTable");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide paging option

By using the [showRowPager](#) and [showColumnPager](#) properties in [pagerSettings](#), you can show or hide row and column pager separately in the pager UI.

In the following example, row pager has been disabled by setting the [showRowPager](#) property in [pagerSettings](#) to **false**.

INDEX.JS


```

var remoteData = new ej.data.DataManager({
    url: 'https://bi.syncfusion.com/northwindservice/api/orders',
    adaptor: new ej.data.WebApiAdaptor(),
    crossDomain: true
});
var pivotObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        type: 'JSON',
        dataSource: remoteData,
        expandAll: true,
        columns: [{
            name: 'ProductName',
            caption: 'Product Name'
        }],
        rows: [{
            name: 'ShipCountry',
            caption: 'Ship Country'
        }, {
            name: 'ShipCity',
            caption: 'Ship City'
        }],
        formatSettings: [{
            name: 'UnitPrice',
            format: 'C0'
        }],
        values: [{
            name: 'Quantity'
        }, {
            name: 'UnitPrice',
            caption: 'Unit Price'
        }],
        filters: []
    },
    width: '100%',
    height: 350,
    enablePaging: true,
    pageSettings: {
        rowPageSize: 10,
        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
    },
    pagerSettings: {
        showRowPager: false
    },
    gridSettings: {
        columnWidth: 120
    },
});
pivotObj.appendTo("#PivotTable");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

[Show or hide page size](#)

By using the [showRowPageSize](#) and [showColumnPageSize](#) properties in [pagerSettings](#), you can show or hide "Rows per page" and "Columns per page" dropdown menu. The dropdown menu contains a list of

pre-defined or user-defined page sizes, which will be displayed in the "Rows per page" and "Columns per page" dropdowns, allowing you to change the page size for the row and column axes at runtime.

INDEX.JS

```
var remoteData = new ej.data.DataManager({
  url: 'https://bi.syncfusion.com/northwindservice/api/orders',
  adaptor: new ej.data.WebApiAdaptor(),
  crossDomain: true
});
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    type: 'JSON',
    dataSource: remoteData,
    expandAll: true,
    columns: [{
      name: 'ProductName',
      caption: 'Product Name'
    }],
    rows: [{
      name: 'ShipCountry',
      caption: 'Ship Country'
    }, {
      name: 'ShipCity',
      caption: 'Ship City'
    }],
    formatSettings: [{
      name: 'UnitPrice',
      format: 'C0'
    }],
    values: [{
      name: 'Quantity'
    }, {
      name: 'UnitPrice',
      caption: 'Unit Price'
    }],
    filters: []
  },
  width: '100%',
  height: 350,
  enablePaging: true,
  pageSettings: {
    rowPageSize: 10,
    columnPageSize: 5,
    currentColumnPage: 1,
    currentRowPage: 1
  },
  pagerSettings: {
    showColumnPageSize: false,
    showRowPageSize: false
  },
  gridSettings: {
    columnWidth: 120
  },
});
pivotObj.appendTo("#PivotTable");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customize page size

By using the [rowPageSizes](#) and [columnPageSizes](#) properties in [pagerSettings](#), you can specify a set of desired page sizes, which will be displayed in the "Rows per page" and "Columns per page" dropdowns, allowing you to change the page size for the row and column axes at runtime.

By default, the "Rows per page" dropdown have pre-defined page sizes of **10, 50, 100, and 200**, while the "Columns per page" dropdown have pre-defined page sizes of **5, 10, 20, 50, and 100**.

In the following example, the "Rows per page" dropdown is set with user-defined page sizes of **10, 20, 30, 40, and 50** and the "Columns per page" dropdown is set with user-defined page sizes of **5, 10, 15, 20, and 30**.

INDEX.JS

```
var remoteData = new ej.data.DataManager({
  url: 'https://bi.syncfusion.com/northwindservice/api/orders',
  adaptor: new ej.data.WebApiAdaptor(),
  crossDomain: true
});
var pivotObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    type: 'JSON',
    dataSource: remoteData,
    expandAll: true,
    columns: [{
      name: 'ProductName',
      caption: 'Product Name'
    }],
    rows: [{
      name: 'ShipCountry',
      caption: 'Ship Country'
    }, {
      name: 'ShipCity',
      caption: 'Ship City'
    }],
    formatSettings: [{
      name: 'UnitPrice',
      format: 'C0'
    }],
    values: [{
      name: 'Quantity'
    }, {
      name: 'UnitPrice',
      caption: 'Unit Price'
    }],
    filters: []
  },
  width: '100%',
  height: 350,
  enablePaging: true,
  pagerSettings: {
    rowPageSize: 10,
    columnPageSize: 5,
    currentColumnPage: 1,
    currentRowPage: 1
  },
  pagerSettings: {
```

```

        columnPageSizes: [5, 10, 15, 20, 30],
        rowPageSizes: [10, 20, 30, 40, 50]
    },
    gridSettings: {
        columnWidth: 120
    },
});
pivotObj.appendTo("#PivotTable");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Template

The [template](#) property allows to change the appearance of the pager UI by displaying user-defined HTML elements instead of built-in HTML elements.

INDEX.JS

```

var remoteData = new ej.data.DataManager({
    url: 'https://bi.syncfusion.com/northwindservice/api/orders',
    adaptor: new ej.data.WebApiAdaptor(),
    crossDomain: true
});
var rowPager;
var columnPager;
var pivotObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        type: 'JSON',
        dataSource: remoteData,
        expandAll: true,
        columns: [{
            name: 'ProductName',
            caption: 'Product Name'
        }],
        rows: [{
            name: 'ShipCountry',
            caption: 'Ship Country'
        }, {
            name: 'ShipCity',
            caption: 'Ship City'
        }],
        formatSettings: [{
            name: 'UnitPrice',
            format: 'C0'
        }],
        values: [{
            name: 'Quantity'
        }, {
            name: 'UnitPrice',
            caption: 'Unit Price'
        }],
        filters: []
    },
    width: '100%',
    height: 350,
    enablePaging: true,
    pageSettings: {
        rowPageSize: 10,
    }
});

```

```

        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
    },
    pagerSettings: {
        template: '#template'
    },
    gridSettings: {
        columnWidth: 120
    },
    dataBound: function () {
        updateTemplate();
    }
});
pivotObj.appendTo("#PivotTable");
function updateTemplate() {
    if (!ej.base.isNullOrUndefined(rowPager)) {
        rowPager.destroy();
        rowPager = null;
    }
    rowPager = new ej.grids.Pager({
        pageSize: pivotObj.pageSettings.rowPageSize,
        totalRecordsCount: pivotObj.engineModule.rowCount,
        currentPage: pivotObj.pageSettings.currentRowPage,
        pageCount: 5,
        click: rowPageClick
    });
    rowPager.appendTo('#row-pager');
    if (!ej.base.isNullOrUndefined(columnPager)) {
        columnPager.destroy();
        columnPager = null;
    }
    columnPager = new ej.grids.Pager({
        pageSize: pivotObj.pageSettings.columnPageSize,
        totalRecordsCount: pivotObj.engineModule.columnCount,
        currentPage: pivotObj.pageSettings.currentColumnPage,
        pageCount: 5,
        click: columnPageClick
    });
    columnPager.appendTo('#column-pager');
}
function rowPageClick(args) {
    pivotObj.pageSettings.currentRowPage = args.currentPage;
    pivotObj.refreshData();
}
function columnPageClick(args) {
    pivotObj.pageSettings.currentColumnPage = args.currentPage;
    pivotObj.refreshData();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```



```

<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <script id="template" type="text/x-template">
    <div class="pager-label">Row Pager: </div>
    <div id="row-pager" class="e-pagertemplate"></div>
    <div class="pager-label">Column Pager: </div>
    <div id="column-pager" class="e-pagertemplate"></div>
  </script>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

State persistence in EJ2 JavaScript Pivotview control

State persistence allows user to maintain the current state of the component along with its report bounded in the browser local storage (cookie). Even if the browser is refreshed or if you move to the next page within the browser, components state will be persisted. State persistence stores the Pivot Table object in the local storage when [enablePersistence](#) property in pivot table is set to **true**.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: [],
  },
  height: 350,
  enablePersistence: true
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Save and Load Pivot Layout

You can save the current layout of the pivot table by using `getPersistData` in string format. The saved layout can be loaded to pivot table any time by passing the saved data as a parameter to `loadPersistData` method in the pivot table.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: [],
    },
    height: 350,
});
pivotTableObj.appendTo('#PivotTable');
var pivotLayout;
var btn = new ej.buttons.Button({ isPrimary: true });
btn.appendTo('#Save');
document.getElementById('Save').addEventListener('click', () => {
    pivotLayout = pivotTableObj.getPersistData();
});
var btn = new ej.buttons.Button({ isPrimary: true });
btn.appendTo('#Load');
document.getElementById('Load').addEventListener('click', () => {
    pivotTableObj.loadPersistData(pivotLayout);
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="PivotTable"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD012 -->

Row and column in EJ2 JavaScript Pivotview control

Width and Height

Allows end user to set the pivot table's height and width by using [height](#) and [width](#) properties in pivot table respectively. The supported formats to set [Height](#) and [Width](#) properties are,

- Pixel: For example - 100, 200, "100px", "200px".
- Percentage: For example - "100%", "200%".
- Auto: It is applicable for [Height](#) property alone in-order to render the pivot table beyond its parent container height without vertical scrollbar. The parent container here would show its vertical scrollbar as soon as the component reaches beyond its dimension.

The pivot table will not be displayed less than **400px**, since it's the minimum width of the component.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  width: 550,
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Row Height

Allows end user to set the height of each pivot table rows commonly using the [rowHeight](#) property in [gridSettings](#).

By default, the [rowHeight](#) property is set as **36** pixels for desktop layout and **48** pixels for mobile layout.

The height of the column headers alone may vary when grouping bar feature is enabled.

In the below code sample, the [rowHeight](#) property is set as **60** pixels.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    gridSettings: {

```

```

        rowHeight: 60
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column Width

Allows end user to set the width of each pivot table columns commonly using the [ColumnWidth](#) property in [gridSettings](#).

By default, the [columnWidth](#) property is set as **110** pixels to each columns except the first column. For first column, **250** pixels and **200** pixels are set respectively with and without grouping bar.

In the below example, the [columnWidth](#) property is set as **120** pixels.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  gridSettings: {
    columnWidth: 120
  }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adjust width based on columns

By default, if the component width set in code-behind is more than the width of the total columns, then the columns will be stretched to make it fit. To avoid the stretching, set the [allowAutoResizing](#) property in the [gridSettings](#) to **false**. By doing so, the component will be adjusted (shrunk) based on the width of total columns.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],

```

```

        values: [{ name: 'Sold', caption: 'Units Sold' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filterSettings: [{ name: 'Year', type: 'Exclude', items: ['FY 2015',
'FY 2017'] }],
        filters: []
    },
    height: 350,
    width: 800,
    gridSettings: {
        columnWidth: 120,
        allowAutoResizing: false
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Reorder

Allows end user to reorder a particular column header from one index to another index within the pivot table through drag-and-drop option. It can be enabled by setting the [allowReordering](#) property in [gridSettings](#) to **true**.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    gridSettings: {
        allowReordering: true
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column Resizing

Allows end user to resize the columns by clicking and dragging the right edge of the column header. While dragging, the width of the respective column will be resized immediately. To enable column resizing option, set the [allowResizing](#) property in [gridSettings](#) to **true**.

By default, the column resizing option is enabled.

In RTL mode, user can click and drag the left edge of the header cell to resize the column.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  gridSettings: {
    allowResizing: true
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

```

```

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Text Wrap

Allows end user to wrap the cell content to the next line when it exceeds the boundary of the cell width. To enable text wrap, set the [allowTextWrap](#) property in [gridSettings](#) to **true**.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    gridSettings: {
        allowTextWrap: true
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Pivot Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Text Align

Allows end user to align the content of the pivot table's row and column headers and value cells by using the [textAlign](#) and [headerTextAlign](#) properties in the [columnRender](#) event under [gridSettings](#). The following alignments are:

- **Left** - It allows the content to be positioned on the left.
- **Right** - It allows the content to be positioned on the right.
- **Center** - It allows the content to be positioned in the middle.
- **Justify** - It allows the content to be as flexible as possible, when the cell does not occupy the entire available area.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  gridSettings: {
    columnRender: function(args) {
      if(args.stackedColumns[0]){
        // Content for the row headers is right-aligned here.
        args.stackedColumns[0].textAlign="Right";
      }
      if(args.stackedColumns[1]){
        // Content for the column header "FY 2015" is center-aligned
here.
        args.stackedColumns[1].textAlign = 'Center';
      }
      if(args.stackedColumns[1] && args.stackedColumns[1].columns[0]){
        // Content for the column header "Q1" is right-aligned here.
        args.stackedColumns[1].columns[0].textAlign = 'Right';
      }
      if(args.stackedColumns[1] && args.stackedColumns[1].columns[0]
&& args.stackedColumns[1].columns[0].columns[0]){
        // Content for the value header "Units Sold" is right-
aligned here.
        args.stackedColumns[1].columns[0].columns[0].headerTextAlign
= 'Right';
      }
      if(args.stackedColumns[1] && args.stackedColumns[1].columns[0]
&& args.stackedColumns[1].columns[0].columns[0]){
        // Content for the values are left-aligned here.
        args.stackedColumns[1].columns[0].columns[0].textAlign =
'Left';
      }
    }
  }
});
```



```

    },
    height: 350
  });
  pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

AutoFit

Allows the user to fit the Pivot Table columns as wide as the content of the cell without wrapping. It auto fits all of the Pivot Table columns by invoking the [autoFitColumns](#) method from the grid instance.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
    enableSorting: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  width: 800,
  dataBound: function (args) {
    pivotTableObj.grid.autoFitColumns();
  },
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The minimum width of 250 pixels is set by default with the grouping bar UI for the first column and cannot be reduced further. So, when the grouping bar is enabled, one can auto fit the Pivot Table columns by calling the [autoFitColumns](#) method from the grid instance with the parameter contained pivot table columns field name excluding first column.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },

```

```

height: 350,
width: 800,
showGroupingBar: true,
dataBound: function (args) {
    var columns = [];
    for (var i = 1; i < pivotTableObj.grid.columnModel.length; i++) {
        columns.push(pivotTableObj.grid.columnModel[i].field);
    }
    pivotTableObj.grid.autoFitColumns(columns);
},
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div>
    <div id="PivotTable"></div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Autofit specific columns

During initial rendering, the parameter `autoFit` in the [columnRender](#) event under [gridSettings](#) can be set to `true` to auto fit specific columns.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
    enableSorting: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  width: 800,
  gridSettings: {
    columnRender: function (args) {
      for (var i = 0; i < args.columns.length; i++) {
        args.columns[i].autoFit = true;
      }
    }
  }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid Lines

Allows end user to display cell border for each cells using [gridLines](#) property in [gridSettings](#).

Available mode of grid lines are:

| Modes | Actions |

|-----|-----|

| Both | Displays both the horizontal and vertical grid lines.|

| None | No grid lines are displayed.|

| Horizontal | Displays the horizontal grid lines only.|

| Vertical | Displays the vertical grid lines only.|

| Default | Displays grid lines based on the theme.|

By default, pivot table renders grid lines in **Both** mode.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  gridSettings: {
    gridLines: 'Vertical'
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection

Selection provides an option to highlight a row or a column or a cell. It can be done through simple mouse down or arrow keys. To enable selection in the pivot table, set the [allowSelection](#) property in [gridSettings](#) to **true**.

The pivot table supports two types of selection that can be set using [type](#) property in [selectionSettings](#). The selection types are:

- **Single:** It is set by default, and it only allows selection of a single row or a column or a cell.
- **Multiple:** Allows you to select multiple rows or columns or cells.

To perform multi-selection, press and hold "CTRL" key and click the desired rows or cells. To select range of rows or cells, press and hold the "SHIFT" key and click the rows or columns or cells.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,

```



```

        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    gridSettings: {
        allowSelection: true,
        selectionSettings: { type: 'Multiple' }
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection Mode

The pivot table supports four types of selection mode that can be set using [mode](#) in [selectionSettings](#). The selection modes are:

- **Row:** It is set by default, and allows user to select only rows.
- **Column:** Allows you to select only columns.
- **Cell:** Allows you to select only cells.
- **Both:** Allows you to select rows and columns at the same time.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    gridSettings: {
        allowSelection: true,
        selectionSettings: { mode: 'Both' }
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Cell Selection Mode

The pivot table supports two types of cell selection mode that can be set using [cellSelectionMode](#) in [selectionSettings](#). The cell selection modes are:

- **Flow:** The **Flow** value is set by default. The range of cells are selected between the start index and end index that includes in between cells of rows.
- **Box:** Range of cells are selected from the start and end column indexes that includes in between cells of rows within the range.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  gridSettings: {
    allowSelection: true,
    selectionSettings: { cellSelectionMode: 'Box', type: 'Multiple',
mode: 'Cell' }
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell selection requires `selectionSettings.mode` property to be `Cell` or `Both`, and `selectionSettings.type` property should be `Multiple`.

Changing background color of the selected cell

The background-color of the selected cell can be changed using built-in CSS names. To do so, please refer to the code sample below, which shows that the selected cells are changed to a **green yellow** color.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
    }
});

```

```

        filters: []
    },
    height: 350,
    gridSettings: {
        allowSelection: true,
        selectionSettings: { cellSelectionMode: 'Box', type: 'Multiple',
mode: 'Cell' }
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>

```

```

        <div id="PivotTable"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Event

CellSelected

The event **cellSelected** is triggered when cell selection gets completed. It provides selected cells information with its corresponding column and row headers. It has following parameters - **selectedCellsInfo**, **currentCell** and **target**. This event allows user to view selected cells information and user can pass those selected cells information to any external component for data binding.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    gridSettings: {
        allowSelection: true,
        selectionSettings: { cellSelectionMode: 'Box', type: 'Multiple',
mode: 'Cell' }
    },
    cellSelected: function(args)
    {
        //args.SelectedCellsInfo -> get selected cells information
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Pivot Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

CellSelecting

The event `cellSelecting` triggers before cell gets selected gets completed. It provides selected cells information with its corresponding column and row headers. It has following parameters - `currentCell`, `data` and `cancel`.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  gridSettings: {
    allowSelection: true,
    selectionSettings: { cellSelectionMode: 'Box', type: 'Multiple',
mode: 'Cell' }
  },
  cellSelecting: function(args)
  {
    //args.currentCell -> get current selected cells information
  }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clip Mode

The clip mode provides options to display its overflow cell content in the pivot table. It can be configured using the [clipMode](#) property in [gridSettings](#). The pivot table supports three types of clip modes which are:

- **Clip:** Truncates the cell content when it overflows its area.
- **Ellipsis:** Displays ellipsis when the cell content overflows its area.
- **EllipsisWithTooltip:** Displays ellipsis when the cell content overflows its area, also it will display the tooltip while hover on ellipsis is applied.

By default, `clipMode` value is set to `Ellipsis`.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],

```

```

        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    gridSettings: {
        clipMode: 'Clip'
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>

```

```

        <div id="PivotTable"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell Template

User can customize the pivot table cell element by using the `cellTemplate` property in pivot table. The `cellTemplate` property accepts either an HTML string or the element's ID, which can be used to append additional HTML elements to showcase each cell with custom format.

In this demo, the revenue cost for each year is represented with trend icons.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: renewableEnergy,
        expandAll: true,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015', 'FY 2017', 'FY
2018'] }],
        formatSettings: [{ name: 'ProCost', format: 'C0' }],
        rows: [
            { name: 'Year', caption: 'Production Year' }
        ],
        columns: [
            { name: 'EnerType', caption: 'Energy Type' },
            { name: 'EneSource', caption: 'Energy Source' }
        ],
        values: [
            { name: 'ProCost', caption: 'Revenue Growth' }
        ],
        filters: []
    },
    gridSettings: {
        columnWidth: 140
    },
    height: 350,
    cellTemplate: '<span class="tempwrap sb-icon-neutral pv-icons"></span>',
    dataBound: function (args) {
        var cTable = document.getElementsByClassName("e-table");
        var colLen = pivotTableObj.pivotValues[3].length;
        var cLen = cTable[3].children[0].children.length;
        var rLen = cTable[3].children[1].children.length;

        for (let k = 0; k < rLen; k++) {
            if (pivotTableObj.pivotValues[k] &&
pivotTableObj.pivotValues[k][0] !== undefined) {
                rowIndx = (pivotTableObj.pivotValues[k][0]).rowIndex;
            }
        }
    }
});

```

```

        break;
    }
}
var rowHeaders =
[[]].slice.call(cTable[2].children[1].querySelectorAll('td'));
var rows = pivotTableObj.dataSourceSettings.rows;
if (rowHeaders.length > 1) {
    for (var i = 0, Cnt = rows; i < Cnt.length; i++) {
        var fields = {};
        var fieldHeaders = [];
        for (var j = 0, Lnt = rowHeaders; j < Lnt.length; j++) {
            var header = rowHeaders[j];
            if (header.className.indexOf('e-gtot') === -1 &&
header.className.indexOf('e-rowsheader') > -1 &&
header.getAttribute('fieldname') === rows[i].name) {
                var headerName =
rowHeaders[j].getAttribute('fieldname') + '_' + rowHeaders[j].textContent;
                fields[rowHeaders[j].textContent] = j;
                fieldHeaders.push(rowHeaders[j].textContent);
            }
        }
        if (i === 0) {
            for (var rnt = 0, Lnt = fieldHeaders; rnt <
Lnt.length; rnt++) {
                if (rnt !== 0) {
                    var row = fields[fieldHeaders[rnt]];
                    var prevRow = fields[fieldHeaders[rnt - 1]];
                    for (var j = 0, ci = 1; j < cLen && ci <
colLen; j++, ci++) {
                        var node =
cTable[3].children[1].children[row].childNodes[j];
                        var prevNode =
cTable[3].children[1].children[prevRow].childNodes[j];
                        var ri = node.getAttribute('index');
                        var prevRi =
prevNode.getAttribute('index');
                        if (ri <
pivotTableObj.pivotValues.length) {
                            if
((pivotTableObj.pivotValues[prevRi][ci]).value >
(pivotTableObj.pivotValues[ri][ci]).value &&
node.querySelector('.tempwrap')) {
                                var trendElement =
node.querySelector('.tempwrap');
                                trendElement.className =
trendElement.className.replace('sb-icon-neutral', 'sb-icon-loss');
                            } else if
((pivotTableObj.pivotValues[prevRi][ci]).value <
(pivotTableObj.pivotValues[ri][ci]).value &&
node.querySelector('.tempwrap')) {
                                var trendElement =
node.querySelector('.tempwrap');
                                trendElement.className =
trendElement.className.replace('sb-icon-neutral', 'sb-icon-profit');
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```
}  
}  
}  
}  
}  
});  
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
</body>
</html>
```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Events

QueryCellInfo

The event `queryCellInfo` triggers while rendering each row and value cells in the pivot table. It allows the user to customize the current cell like adding or removing styles, editing value, etc. It has the following parameters:

- `cell` - It holds the current cell information.
- `data` - It holds the entire row data across the current cell.
- `column` - It holds the entire column data across the current cell.
- `pivotview` - It holds pivot table instance.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    gridSettings: {
        columnWidth: 120,
        queryCellInfo: function (args) {
            // triggers every time for value cell while rendering
        },
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

HeaderCellInfo

The event `headerCellInfo` triggers while rendering each column header cell in the pivot table. It allows the user to customize the element of the current header cell like adding or removing styles, editing value, etc. It has the following parameters:

- **node** - It holds the current header cell information

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  gridSettings: {
    columnWidth: 120,
    headerCellInfo: function (args) {
      // triggers every time for header cell while rendering
    },
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ColumnRender

The event [columnRender](#) triggers while framing each columns for rendering in the pivot table. It allows the user to customize the text alignment, column visibility, autofit, re-ordering, minimum and maximum width for a specific column. It has the following parameters:

- **columns** - It holds the leaf level columns (i.e., value headers) information.
- **dataSourceSettings** - It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **stackedColumns** - It holds the drilled columns (i.e., including column and value headers) information.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
    }
});

```

```

        filters: []
    },
    gridSettings: {
        columnRender: function (args) {
            // Here you can customize the specific columns.
            for (var i = 0; i < args.columns.length; i++) {
                args.columns[i].autoFit = true;
                args.columns[i].textAlign = "Right";
            }
        },
        height: 350
    });
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

CellClick

The event [cellClick](#) triggers while clicking a cell in the pivot table. For instance, using this event end-user can either add or remove styles, edit value and also perform any other DOM manipulations. It has the following parameters:

- **currentCell** - It holds the current cell information.
- **data** - It holds the clicked cell's data like axis, formatted text, actual text, row header, column header and value informations.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  gridSettings: {
    allowSelection: true,
    selectionSettings: { cellSelectionMode: 'Box', type: 'Multiple',
mode: 'Cell' }
  },
  cellClick: function(args)
  {
    args.currentCell.setAttribute("style", "background-color: red;")
  }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Summary customization in EJ2 JavaScript Pivotview control

Show or hide grand totals

Allows to show or hide grand totals in rows and columns using the [showGrandTotals](#) property. To hide the grand totals in rows and columns, set the property [showGrandTotals](#) in [dataSourceSettings](#) to **false**.

End user can also hide grand totals for row or columns separately by setting the property [showRowGrandTotals](#) or [showColumnGrandTotals](#) in [dataSourceSettings](#) to **false** respectively.

By default, [showGrandTotals](#), [showRowGrandTotals](#) and [showColumnGrandTotals](#) properties in [dataSourceSettings](#) are set as **true**.

INDEX.JS

```
var pivotGridObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: [],
    showGrandTotals: false
  },
  height: 350,
  showFieldList: true
});
pivotGridObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show grand totals at top or bottom

Allows to show grand totals either at top or bottom in rows and columns using the [grandTotalsPosition](#) property. To show the grand totals at top in rows and columns, set the [grandTotalsPosition](#) property in [dataSourceSettings](#) to **Top**.

INDEX.JS

```

var pivotGridObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        grandTotalsPosition: 'Top',
    },
    height: 350,
});
pivotGridObj.appendTo('#PivotTable');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```


Show or hide sub-totals

Allows to show or hide sub-totals in rows and columns using the [showSubTotals](#) property. To hide all the sub-totals in rows and columns, set the property [showSubTotals](#) in [dataSourceSettings](#) to **false**. End user can also hide sub-totals for rows or columns separately by setting the property [showRowSubTotals](#) or [showColumnSubTotals](#) in [dataSourceSettings](#) to **false** respectively.

By default, [showSubTotals](#), [showRowSubTotals](#) and [showColumnSubTotals](#) properties in [dataSourceSettings](#) are set as **true**.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: [],
    showSubTotals: false
  },
  height: 350,
  showFieldList: true
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide sub-totals for specific fields

Allows to show or hide sub-totals for specific fields in rows and columns using the [showSubTotals](#) property. To hide sub-totals for a specific field in row or column axis, set the property [showSubTotals](#) in [rows](#) or [columns](#) to **false** respectively.

By default, [showSubTotals](#) property in [rows](#) or [columns](#) is set as **true**.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year', showSubTotals:
false }, { name: 'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country', showSubTotals: false }, { name: 'Products'
}],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },

```

```

        height: 350,
        showFieldList: true
    });
    pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show sub-totals at top or bottom

Allows to show sub-totals either at top or bottom of the header group in rows and columns by using the [subTotalsPosition](#) property. By default, [subTotalsPosition](#) property is set to **Auto**, which means that column sub-totals are displayed at the bottom and row sub-totals are displayed at the top of the header group in the pivot table.

To show sub-totals at top of the header group in rows and columns, set the [subTotalsPosition](#) property in [dataSourceSettings](#) to **Top**.

INDEX.JS

```

var pivotGridObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    drilledMembers: [{ name: 'Country', items: ['France'] }, { name:
'Year', items: ['FY 2015'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    subTotalsPosition: 'Top'
  },
  height: 350,
  showFieldList: true
});
pivotGridObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To show sub-totals at bottom of the header group in rows and columns, set the [subTotalsPosition](#) property in [dataSourceSettings](#) to **Bottom**.

INDEX.JS

```

var pivotGridObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }, { name:
'Year', items: ['FY 2015'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        subTotalsPosition: 'Bottom'
    },

```

```

        height: 350,
        showFieldList: true
    });
    pivotGridObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide totals using toolbar

It can also be achieved using built-in toolbar options by setting the [showToolbar](#) property in pivot table to **true**. Also, include the items **GrandTotal** and **SubTotal** within the [toolbar](#) property in pivot table. End user can now see "Show/Hide Grand totals" and "Show/Hide Sub totals" icons in toolbar UI automatically.

The grand totals and sub-totals can be dynamically displayed at the top or bottom of the pivot table's row and column axes by using the built-in options "Grand totals position" and "Subtotals position" available in the grand totals and sub-totals drop down menus, respectively.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France', 'Germany']
  }],
  columns: [{ name: 'Year' }],
  rows: [{ name: 'Country' }, { name: 'Products' }],
  values: [{ name: 'Amount', caption: 'Sold Amount' },
    { name: 'Sold', caption: 'Units Sold' }],
  filters: []
},
  showToolbar: true,
  toolbar: ['GrandTotal', 'SubTotal'],
  height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hyper link in EJ2 JavaScript Pivotview control

The pivot table supports to show hyperlink option to link data for individual cells that are displayed in the component. Also, the hyperlink can be enabled separately for row headers, column headers, value cells, and summary cells using the [hyperlinkSettings](#). It can be configured through code behind, during initial rendering and the settings available to show hyperlink are:

- [showHyperlink](#): It allows to set the visibility of hyperlink in all cells.
- [showRowHeaderHyperlink](#): It allows to set the visibility of hyperlink in row headers.
- [showColumnHeaderHyperlink](#): It allows to set the visibility of hyperlink in column headers.
- [showValueCellHyperlink](#): It allows to set the visibility of hyperlink in value cells.
- [showSummaryCellHyperlink](#): It allows to set the visibility of hyperlink in summary cells.
- [headerText](#): It allows to set the visibility of hyperlink based on header text.
- [conditionalSettings](#): It allows to set the visibility of hyperlink based on specific condition.

<!-- markdownlint-disable MD028 -->

By default, the hyperlink options are disabled for all cells in the pivot table.

User defined style can be applied to hyperlink using [cssClass](#) property in [hyperlinkSettings](#).

Hyperlink for all cells

The pivot table has an option to show hyperlink option to all the cells that are currently displaying. To do so, user need to set [showHyperlink](#) to **true**.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  hyperlinkSettings: {
    showHyperlink: true,
    cssClass: 'e-custom-class'
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hyperlink for row headers

The pivot table has an option to show hyperlink option for row header cells alone that are currently in display. To do so, user need to set [showRowHeaderHyperlink](#) to **true**.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    hyperlinkSettings: {

```

```

        showRowHeaderHyperlink: true,
        cssClass: 'e-custom-class'
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hyperlink for column headers

The pivot table has an option to show hyperlink option for column header cells alone that are currently in display. To do so, user need to set [showColumnHeaderHyperlink](#) to **true**.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    hyperlinkSettings: {
        showColumnHeaderHyperlink: true,
        cssClass: 'e-custom-class'
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hyperlink for value cells

The pivot table has an option to show hyperlink option for value cells alone that are currently in display. To do so, user need to set [showValueCellHyperlink](#) to **true**.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],

```

```

        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    hyperlinkSettings: {
        showValueCellHyperlink: true,
        cssClass: 'e-custom-class'
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>

```

```

        <div id="PivotTable"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hyperlink for summary cells

The pivot table has an option to show hyperlink option for summary cells alone that are currently in display. To do so, user need to set [showSummaryCellHyperlink](#) to **true**.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    hyperlinkSettings: {
        showSummaryCellHyperlink: true,
        cssClass: 'e-custom-class'
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Condition based hyperlink

The pivot table has an option to show hyperlink option to the cells based on specific conditions. It can be configured using the [conditionalSettings](#) option through code behind, during initial rendering. The required settings are:

- [measure](#): Specifies the value field caption to get visibility of hyperlink option for specific measure.
- [condition](#): Specifies the operator type such as equals, greater than, less than, etc.
- [value1](#): Specifies the start value.
- [value2](#): Specifies the end value.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  hyperlinkSettings: {
    conditionalSettings: [{
      measure: 'Sold',
      conditions: 'Between',
      value1: 150,
      value2: 200
    }],
    cssClass: 'e-custom-class'
  }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Header based hyperlink

The pivot table has an option to show hyperlink in the cells based on specific row or column header. It can be configured using the [headerText](#) option through code behind, during initial rendering.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    hyperlinkSettings: {
        headerText: 'FY 2015.Q1.Units Sold',

```

```

        cssClass: 'e-custom-class'
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Event

The event [hyperlinkCellClicked](#) fires on every hyperlink cell click.

It has following parameters - `cancel` and `currentCell`. The parameter `currentCell` is used to customize the host cell element by any means. Meanwhile, when the parameter `cancel` is set to `true`, applied customization will not be updated to the host cell element.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  hyperlinkSettings: {
    showHyperlink: true,
    cssClass: 'e-custom-class'
  },
  hyperlinkCellClick: function (args) {
    args.cancel = false;
    args.currentCell.setAttribute("data-url",
"https://ej2.syncfusion.com/");//here we have redirected to EJ2 Syncfusion
on hyperlinkcell click
  }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Apply condition based hyperlink for specific row or column](#)

Tool bar in EJ2 JavaScript Pivotview control

Toolbar option allows to access the frequently used features like switching between pivot table and pivot chart, changing chart types, conditional formatting, exporting, etc... with ease at runtime. This

option can be enabled by setting the [showToolbar](#) property in pivot table to **true**. The [toolbar](#) property in pivot table accepts the collection of built-in toolbar options.

Built-in Toolbar Options

The following table shows built-in toolbar options and its actions.

Built-in Toolbar Options	Actions
----- -----	
New	Creates a new report
Save	Saves the current report
Save As	Save as current report
Rename	Renames the current report
Delete	Deletes the current report
Load	Loads any report from the report list
Grid	Shows pivot table
Chart	Shows a chart in any type from the built-in list and option to enable/disable multiple axes
Exporting	Exports the pivot table as PDF/Excel/CSV and the pivot chart as PDF and image
Sub-total	Shows or hides sub totals
Grand Total	Shows or hides grand totals
Conditional Formatting	Shows the conditional formatting pop-up to apply formatting
Number Formatting	Shows the number formatting pop-up to apply number formatting
Field List	Shows the fieldlist pop-up
MDX	Shows the MDX query that was run to retrieve data from the OLAP data source. NOTE: This applies only to the OLAP data source.

The order of toolbar options can be changed by simply moving the position of items in the toolbar collection. Also if end user wants to remove any toolbar option from getting displayed, it can be simply ignored from adding into the toolbar collection.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    sortSettings: [{ name: 'Country', order: 'Descending' }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: []
  },
},
```

```

height: 350,
saveReport: function (args) {
    var reports = [];
    var isSaved = false;
    if (localStorage.pivotviewReports &&
localStorage.pivotviewReports !== "") {
        reports = JSON.parse(localStorage.pivotviewReports);
    }
    if (args.report && args.reportName && args.reportName !== '') {
        reports.map(function (item) {
            if (args.reportName === item.reportName) {
                item.report = args.report;
                isSaved = true;
            }
        });
        if (!isSaved) {
            reports.push(args);
        }
        localStorage.pivotviewReports = JSON.stringify(reports);
    }
},
fetchReport: function (args) {
    var reportCollection = [];
    var reeportList = [];
    if (localStorage.pivotviewReports &&
localStorage.pivotviewReports !== "") {
        reportCollection =
JSON.parse(localStorage.pivotviewReports);
    }
    reportCollection.map(function (item) {
        reeportList.push(item.reportName);
    });
    args.reportName = reeportList;
},
loadReport: function (args) {
    var reportCollection = [];
    if (localStorage.pivotviewReports &&
localStorage.pivotviewReports !== "") {
        reportCollection =
JSON.parse(localStorage.pivotviewReports);
    }
    reportCollection.map(function (item) {
        if (args.reportName === item.reportName) {
            args.report = item.report;
        }
    });
    if (args.report) {
        pivotTableObj.dataSource =
JSON.parse(args.report).dataSource;
    }
},
removeReport: function (args) {
    var reportCollection = [];
    if (localStorage.pivotviewReports &&
localStorage.pivotviewReports !== "") {
        reportCollection =
JSON.parse(localStorage.pivotviewReports);
    }
}

```

```

        for (var i = 0; i < reportCollection.length; i++) {
            if (reportCollection[i].reportName === args.reportName) {
                reportCollection.splice(i, 1);
            }
        }
        if (localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== "") {
            localStorage.pivotviewReports =
            JSON.stringify(reportCollection);
        }
    },
    renameReport: function (args) {
        var reportCollection = [];
        if (localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== "") {
            reportCollection =
            JSON.parse(localStorage.pivotviewReports);
        }
        reportCollection.map(function (item) { if (args.reportName ===
        item.reportName) {
            item.reportName = args.rename;
        } });
        if (localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== "") {
            localStorage.pivotviewReports =
            JSON.stringify(reportCollection);
        }
    },
    newReport: function () {
        pivotTableObj.setProperties({ dataSourceSettings: { columns: [],
        rows: [], values: [], filters: [] } }, false);
    },
    displayOption: { view: 'Both' },
    chartSettings: {
        value: 'Amount', enableExport: true, chartSeries: { type:
        'Column', animation: { enable: false } }, enableMultipleAxis: false,
    },
    toolbar: ['New', 'Save', 'SaveAs', 'Rename', 'Remove', 'Load',
        'Grid', 'Chart', 'Export', 'SubTotal', 'GrandTotal',
        'ConditionalFormatting', 'FieldList'],
    allowExcelExport: true,
    allowConditionalFormatting: true,
    allowPdfExport: true,
    showToolbar: true,
    allowCalculatedField: true,
    showFieldList: true
    });
    pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">

```



```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show desired chart types in the dropdown menu

By default, all chart types are displayed in the dropdown menu included in the toolbar. However, based on the request for an application, we may need to show selective chart types on our own. This can be achieved using the [chartTypes](#) property. To know more about supporting chart types, [click here](#).

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    enableSorting: true,
    sortSettings: [{ name: 'Country', order: 'Descending' }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Both' },
  toolbar: ['Grid', 'Chart'],
  chartTypes: ['Column', 'Bar', 'Line', 'Area'],
  showToolbar: true,
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">
```

```

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet">

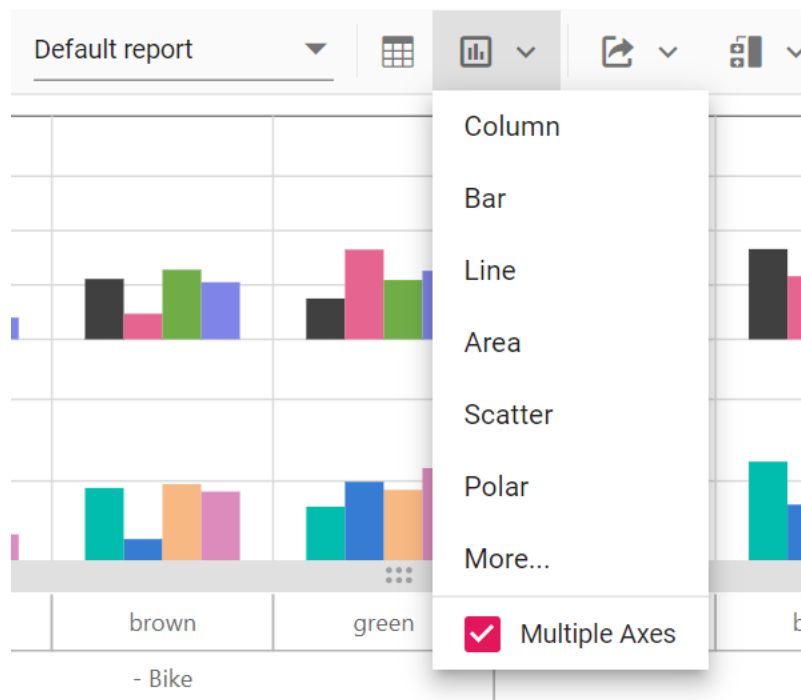
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

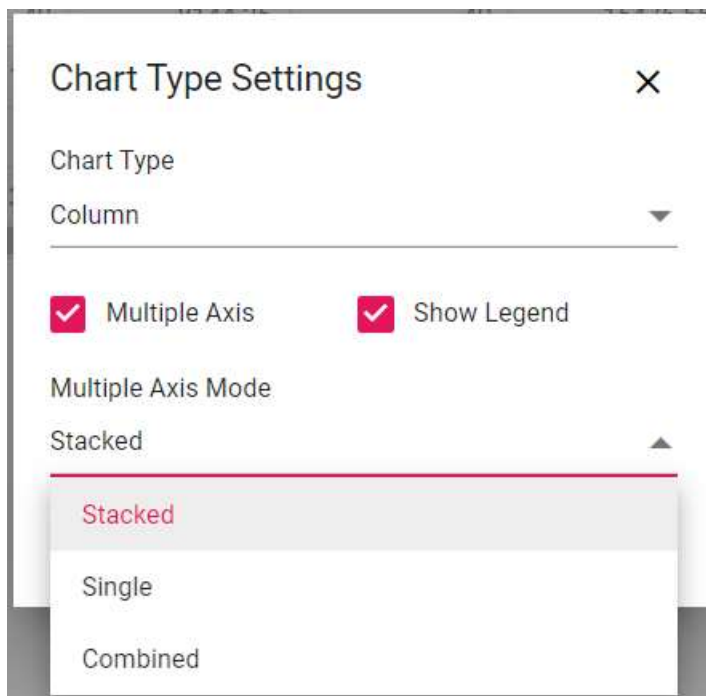
Switch the chart to multiple axes

In the chart, the user can switch from single axis to multiple axes with the help of the built-in checkbox available inside the chart type dropdown menu in the toolbar. For more information [refer here](#).



<!-- markdownlint-disable MD009 -->

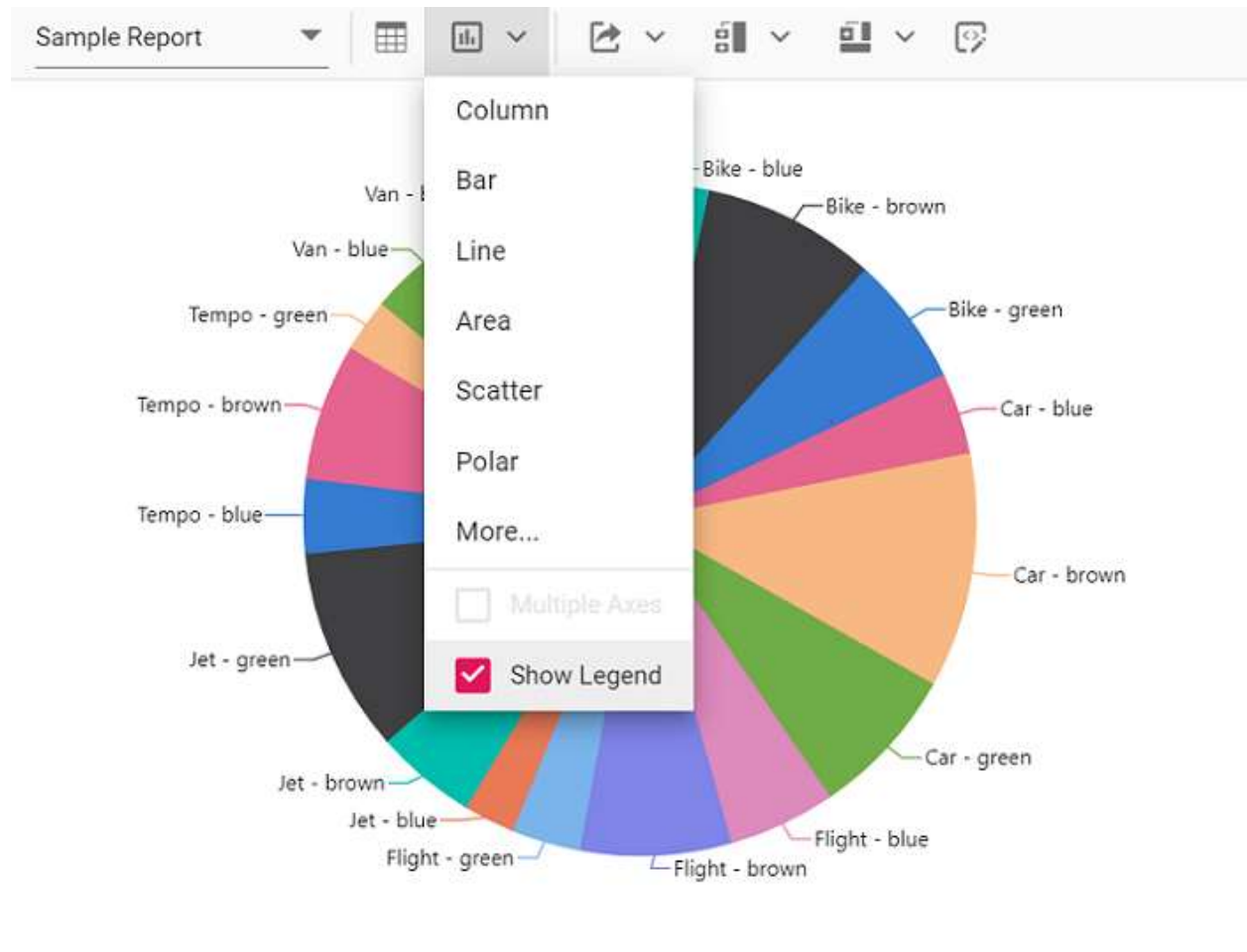
There are three modes available in **Multiple Axis** option: **Stacked**, **Single** and **Combined**. The modes can be changed using “Multiple Axis Mode” drop-down list which appears while clicking the **More...** option.



Show or hide legend

In the chart, legend can be shown or hidden dynamically with the help of the built-in option available in the chart type drop-down menu.

By default, the legend is not visible for the accumulation chart types like pie, doughnut, pyramid, and funnel. Users can enable or disable using the built-in checkbox option.



Adding custom option to the toolbar

In addition to the existing built-in toolbar items, new toolbar item(s) may also be included. This can be achieved by using the [toolbarRender](#) event. The action of the new toolbar item(s) can also be defined within this event.

The new toolbar item(s) can be added to the desired position in the toolbar using the `splice` option.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    sortSettings: [{ name: 'Country', order: 'Descending' }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: []
  },
  height: 350,
  toolbarRender: function (args) {
```

```

        args.customToolbar.splice(12, 0, {
            prefixIcon: 'e-tool-expand e-icons', tooltipText:
'Expand/Collapse',
            click: this.toolbarClicked.bind(this),
        });
    },
    toolbarClicked: function(args) {
        pivotTableObj.dataSourceSettings.expandAll =
!pivotTableObj.dataSourceSettings.expandAll;
    },
    displayOption: { view: 'Both' },
    chartSettings: {
        value: 'Amount', enableExport: true, chartSeries: { type: 'Column',
animation: { enable: false } }, enableMultipleAxis: false,
    },
    toolbar: ['Expand/Collapse'],
    showToolbar: true,
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In the above topic, we have seen how to add an icon as one of the toolbar item in toolbar panel. In the next topic, we are going to see how to frame the entire toolbar panel and how to add a custom control in it.

Toolbar Template

It allows to customize the toolbar panel by using template option. It allows any custom control to be used as one of the toolbar item inside the toolbar panel. It can be achieved by two ways,

Here, the entire toolbar panel can be framed in HTML elements that are appended at the top of the pivot table. The **id** of the HTML element needs to be set in the [toolbarTemplate](#) property in-order to map it to the pivot table.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: [],
    },
    toolbarTemplate: '#template',
    showToolbar: true,
    height: 350,
});
pivotTableObj.appendTo('#PivotTable');
var btnExpand = new ej.buttons.Button({ content: 'Expand All', cssClass: "e-
primary" });
btnExpand.appendTo('#btnexpand');

```

```

var btnCollapse = new ej.buttons.Button({ content: 'Collapse All', cssClass:
'e-primary' });
    btnCollapse.appendTo('#btncollapse');
document.getElementById('btnexpand').onclick = function () {
    pivotTableObj.dataSourceSettings.expandAll = true;
};
document.getElementById('btncollapse').onclick = function () {
    pivotTableObj.dataSourceSettings.expandAll = false;
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>

```



```

        <div id="template">
            <div>
                <div> <button id="btnexpand" class="e-flat"></button></div>
                <div> <button id="btncollapse" class="e-
flat"></button></div>
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Another option allows to frame a custom toolbar item using HTML elements and include in the toolbar panel at the desired position. The custom toolbar items can be declared as control **instance** or element **id** in the [toolbar](#) property in pivot table.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: [],
    },
    toolbar: [{template: '#enablertl'}, {template: '#disablertl'}],
    showToolbar: true,
    height: 350,
});
pivotTableObj.appendTo('#PivotTable');
var btnEnableRtl = new ej.buttons.Button({ content: 'Enable Rtl', cssClass:
"e-primary" });
    btnEnableRtl.appendTo('#btnenablertl');
var btnDisableRtl = new ej.buttons.Button({ content: 'Disable Rtl',
cssClass: "e-primary" });
    btnDisableRtl.appendTo('#btndisablertl');
document.getElementById('btnenablertl').onclick = function () {
    pivotTableObj.enableRtl = true;
};
document.getElementById('btndisablertl').onclick = function () {
    pivotTableObj.enableRtl = false;
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
      <div id="enablertl"> <button id="btnenablertl" class="e-
flat"></button></div>
      <div id="disablertl"> <button id="btndisablertl" class="e-
flat"></button></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Note: For both options, the actions for the toolbar template items can be defined in the event [toolbarClick](#). Also, if the toolbar item is a custom control then its built-in events can also be accessed.

```
<!-- markdownlint-disable MD009 -->
```

Save and load report as a JSON file

The current pivot report can be saved as a JSON file in the desired path and loaded back to the pivot table at any time.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: [],
  },
  height: 350,
  dataBound: function (args) {
    var dataSource =
JSON.parse(pivotTableObj.getPersistData()).dataSourceSettings.dataSource;
    var a = document.getElementById('save');
    var mime_type = 'application/octet-stream'; // text/html,
image/png, et c
    a.setAttribute('download', 'pivot.JSON');
    a.href = 'data:' + mime_type + ';base64,' +
btoa(JSON.stringify(dataSource) || '');
    document.getElementById('files').addEventListener('change',
this.readBlob, false);
  },
  readBlob: function (args) {
    var files = document.getElementById('load').files;
    var file = files[0];
    var start = 0;
    var stop = file.size - 1;
    var reader = new FileReader();
    reader.onloadend = function (evt) {
      if (evt.target.readyState == FileReader.DONE) {
        pivotTableObj.dataSource = JSON.parse(evt.target.result);
      }
    };
    var blob = file.slice(start, stop + 1);
    reader.readAsBinaryString(blob);
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
      <a id="save" class="btn btn-primary">Save</a>
      <div class="fileUpload btn btn-primary">
        <span>Load</span>
        <input id="files" type="file" class="upload">
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

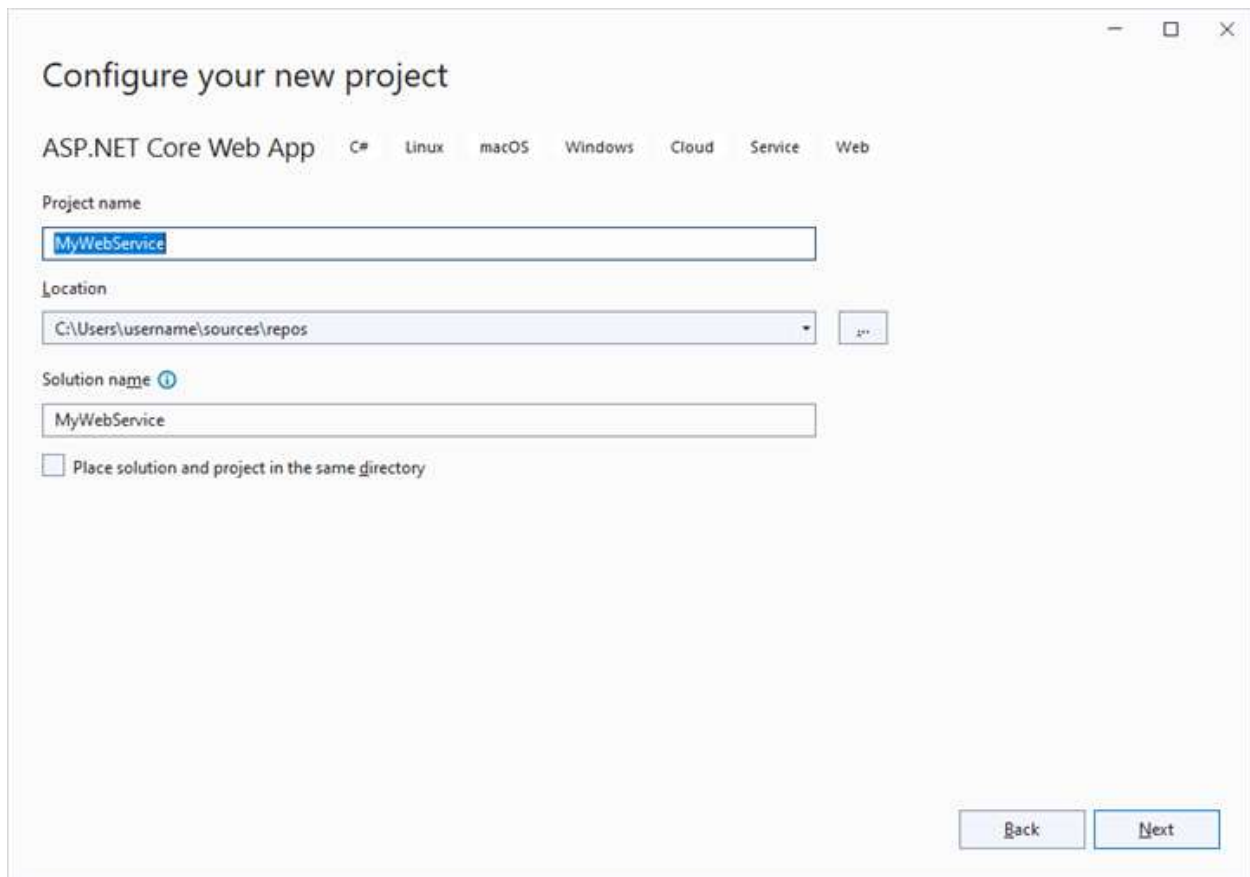
```
    ele.style.visibility = "visible";  
  }  
  </script>  
<script src="index.js" type="text/javascript"></script>  
</body></html>
```

Save and load reports to a SQL database

SQL Server is a relational database management system (RDBMS) that can be used to store and manage large amounts of data. In this topic, we will see how to save, save as, rename, load, delete, and add reports between a SQL Server database and a JavaScript Pivot Table at runtime.

Create a Web API service to connect to a SQL Server database

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).



Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name
MyWebService

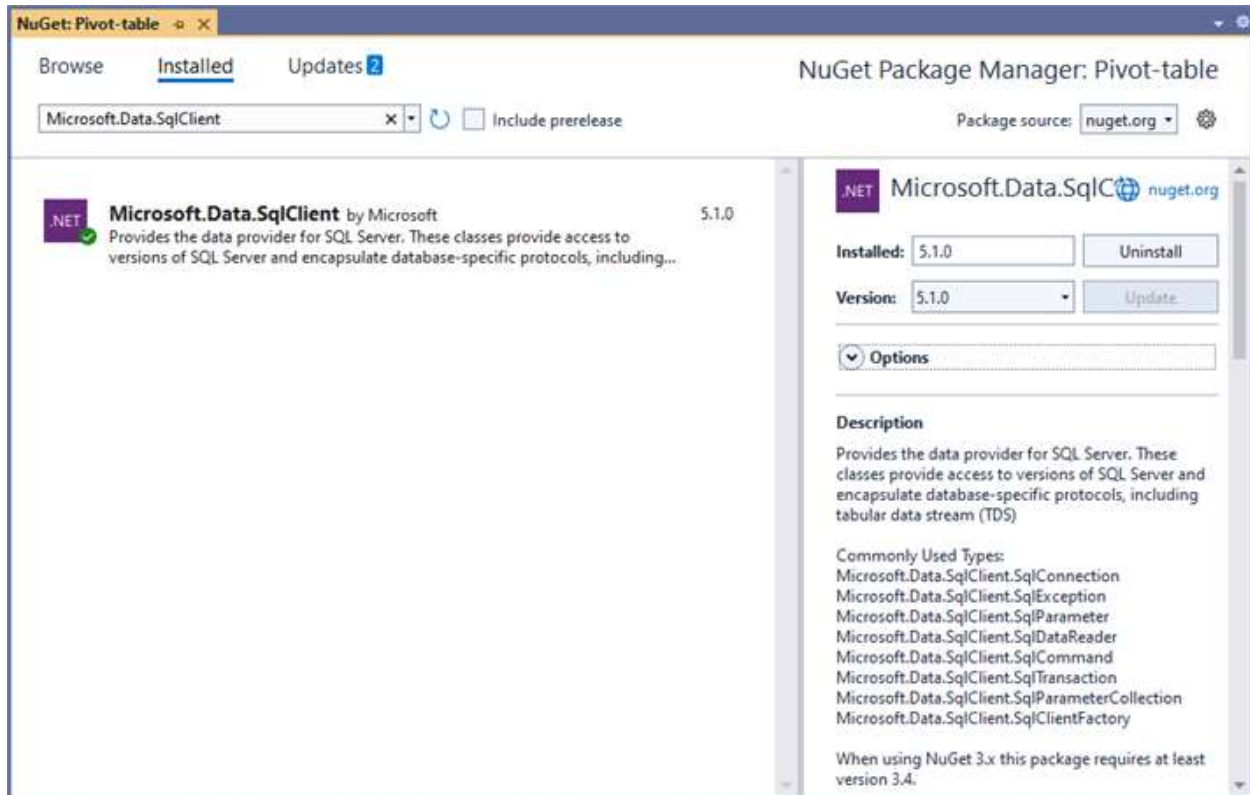
Location
C:\Users\username\sources\repos

Solution name ⓘ
MyWebService

☐ Place solution and project in the same directory

Back Next

2. To connect a SQL Server database using the Microsoft SqlClient in our application, we need to install the [Microsoft.Data.SqlClient](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Microsoft.Data.SqlClient** and install it.



3. Under the **Controllers** folder, create a Web API controller (aka, PivotController.cs) file that aids in data communication with the Pivot Table.

4. In the Web API Controller (aka, PivotController), the **OpenConnection** method is used to connect to the SQL database. The **GetDataTable** method then processes the specified SQL query string, retrieves data from the database, and converts it into a **DataTable** using **SqlCommand** and **SqlDataAdapter**. This **DataTable** can be used to retrieve saved reports and modify them further as shown in the code block below.

[PivotController.cs]

```
`c#
```

```
using Microsoft.AspNetCore.Mvc;
```

```
using Microsoft.Data.SqlClient;
```

```
using System.Data;
```

```
namespace MyWebService.Controllers
```

```
{
```

```
[ApiController]
```

```
[Route("[controller]")]
```

```
public class PivotController : ControllerBase
```

```
{
```

```
[HttpPost]
```

```
[Route("Pivot/SaveReport")]
public void SaveReport([FromBody] Dictionary<string, string> args)
{
    SaveReportToDB(args["ReportName"], args["Report"]);
}

[HttpPost]
[Route("Pivot/FetchReport")]
public string[] FetchReport(Dictionary<string, string> args)
{
    return FetchReportListFromDB().ToArray();
}

[HttpPost]
[Route("Pivot/LoadReport")]
public string LoadReport(Dictionary<string, string> args)
{
    return LoadReportFromDB(args["ReportName"]);
}

[HttpPost]
[Route("Pivot/RemoveReport")]
public void RemoveReport(Dictionary<string, string> args)
{
    RemoveReportFromDB(args["ReportName"]);
}

[HttpPost]
[Route("Pivot/RenameReport")]
public void RenameReport(Dictionary<string, string, bool> args)
{
    RenameReportInDB(args["ReportName"], args["Rename"], args["isReportExists"]);
}

public void SaveReportToDB(string reportName, string report)
{
    SqlConnection sqlConn = OpenConnection();
    bool isDuplicate = true;
```

```
SqlCommand cmd1 = null;
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
    if ((row["ReportName"] as string).Equals(reportName))
    {
        isDuplicate = false;
        cmd1 = new SqlCommand("UPDATE ReportTable set Report=@Report where ReportName like
        @ReportName", sqlConn);
    }
}
if (isDuplicate)
{
    cmd1 = new SqlCommand("INSERT into ReportTable (ReportName,Report)
    Values(@ReportName,@Report)", sqlConn);
}
cmd1.Parameters.AddWithValue("@ReportName", reportName);
cmd1.Parameters.AddWithValue("@Report", report.ToString());
cmd1.ExecuteNonQuery();
sqlConn.Close();
}

public string LoadReportFromDB(string reportName)
{
    SqlConnection sqlConn = OpenConnection();
    string report = string.Empty;
    foreach (DataRow row in GetDataTable(sqlConn).Rows)
    {
        if ((row["ReportName"] as string).Equals(reportName))
        {
            report = (string)row["Report"];
            break;
        }
    }
    sqlConn.Close();
    return report;
}
```



```
}  
public List<string> FetchReportListFromDB()  
{  
    SqlConnection sqlConn = OpenConnection();  
    List<string> reportNames = new List<string>();  
    foreach (DataRow row in GetDataTable(sqlConn).Rows)  
    {  
        if (!string.IsNullOrEmpty(row["ReportName"] as string))  
        {  
            reportNames.Add(row["ReportName"].ToString());  
        }  
    }  
    sqlConn.Close();  
    return reportNames;  
}  
  
public void RenameReportInDB(string reportName, string renameReport, bool isReportExists)  
{  
    SqlConnection sqlConn = OpenConnection();  
    SqlCommand cmd1 = null;  
    if (isReportExists)  
    {  
        RemoveReportFromDB(renameReport);  
    }  
    foreach (DataRow row in GetDataTable(sqlConn).Rows)  
    {  
        if ((row["ReportName"] as string).Equals(reportName))  
        {  
            cmd1 = new SqlCommand("UPDATE ReportTable set ReportName=@RenameReport where ReportName  
like '%" + reportName + "%'", sqlConn);  
            break;  
        }  
    }  
    cmd1.Parameters.AddWithValue("@RenameReport", renameReport);
```

```
cmd1.ExecuteNonQuery();
sqlConn.Close();
}
public void RemoveReportFromDB(string reportName)
{
    SqlConnection sqlConn = OpenConnection();
    SqlCommand cmd1 = null;
    foreach (DataRow row in GetDataTable(sqlConn).Rows)
    {
        if ((row["ReportName"] as string).Equals(reportName))
        {
            cmd1 = new SqlCommand("DELETE FROM ReportTable WHERE ReportName LIKE '%" + reportName +
            "%'", sqlConn);
            break;
        }
    }
    cmd1.ExecuteNonQuery();
    sqlConn.Close();
}
private SqlConnection OpenConnection()
{
    // Replace with your own connection string.
    string connectionString = @"<Enter your valid connection string here>";
    SqlConnection sqlConn = new SqlConnection(connectionString);
    sqlConn.Open();
    return sqlConn;
}
private DataTable GetDataTable(SqlConnection sqlConn)
{
    string xquery = "SELECT * FROM ReportTable";
    SqlCommand cmd = new SqlCommand(xquery, sqlConn);
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
```

```

da.Fill(dt);
return dt;
}
}
}
`

```

5. When you run the app, it will be hosted at <https://localhost:44313>. You can use the hosted URL to save and load reports in the SQL database from the Pivot Table.

Further, let us explore more on how to save, load, rename, delete, and add reports using the built-in toolbar options via Web API controller (aka, PivotController) one-by-one.

Saving a report

When you select the **“Save a report”** option from the toolbar, the [saveReport](#) event is triggered. In this event, an AJAX request is made to the Web API controller's **SaveReport** method, passing the name of the current report and the current report, which you can use to check and save in the SQL database.

For example, the report shown in the following code snippet will be passed to the **SaveReport** method along with the report name **“Sample Report”** and saved in the SQL database.

[index.js]

```

`ts
var pivotTableObj = new ej.pivotview.PivotView({
dataSourceSettings: {
columns: [
{ name: "Year", caption: "Production Year" },
{ name: "Quarter" },
],
dataSource: getPivotData(),
expandAll: false,
filters: [],
formatSettings: [{ name: "Amount", format: "C0" }],
rows: [{ name: "Country" }, { name: "Products" }],
values: [
{ name: "Sold", caption: "Units Sold" },
{ name: "Amount", caption: "Sold Amount" },
],
},
height: 350,

```

```
saveReport: function (args) {
var report = JSON.parse(args.report);
report.dataSourceSettings.dataSource = [];
fetch("https://localhost:44313/Pivot/SaveReport", {
method: "POST",
headers: {
Accept: "application/json",
"Content-Type": "application/json",
},
body: JSON.stringify({
reportName: args.reportName,
report: JSON.stringify(report),
}),
})
.then(function (response) {
pivotTableObj.fetchReport();
});
},
toolbar: ['New', 'Save', 'SaveAs', 'Rename', 'Remove', 'Load',
'Grid', 'Chart', 'Export', 'SubTotal', 'GrandTotal', 'ConditionalFormatting', 'FieldList'],
allowExcelExport: true,
allowConditionalFormatting: true,
allowPdfExport: true,
showToolbar: true,
allowCalculatedField: true,
displayOption: { view: "Both" },
showFieldList: true,
height: "700",
});
pivotTableObj.appendTo("#PivotView");
、

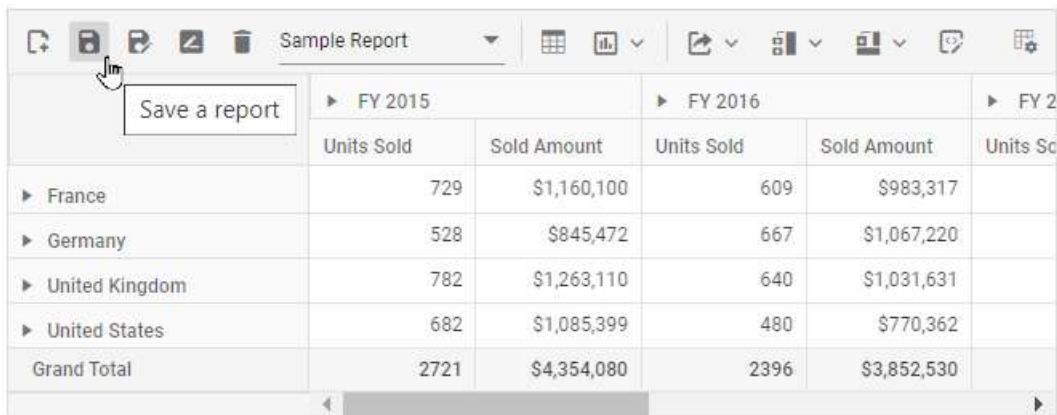
[PivotController.cs]
`c#
```

```
namespace MyWebApp.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class PivotController : ControllerBase
    {
        [HttpPost]
        [Route("Pivot/SaveReport")]
        public void SaveReport(Dictionary<string,string> args)
        {
            SaveReportToDB(args["ReportName"], args["Report"]);
        }
        public void SaveReportToDB(string reportName, string report)
        {
            SqlConnection sqlConn = OpenConnection();
            bool isDuplicate = true;
            SqlCommand cmd1 = null;
            foreach (DataRow row in GetDataTable(sqlConn).Rows)
            {
                if ((row["ReportName"] as string).Equals(reportName))
                {
                    isDuplicate = false;
                    cmd1 = new SqlCommand("UPDATE ReportTable set Report=@Report where ReportName like @ReportName", sqlConn);
                }
            }
            if (isDuplicate)
            {
                cmd1 = new SqlCommand("INSERT into ReportTable (ReportName,Report) Values(@ReportName,@Report)", sqlConn);
            }
            cmd1.Parameters.AddWithValue("@ReportName", reportName);
            cmd1.Parameters.AddWithValue("@Report", report.ToString());
            cmd1.ExecuteNonQuery();
        }
    }
}
```

```

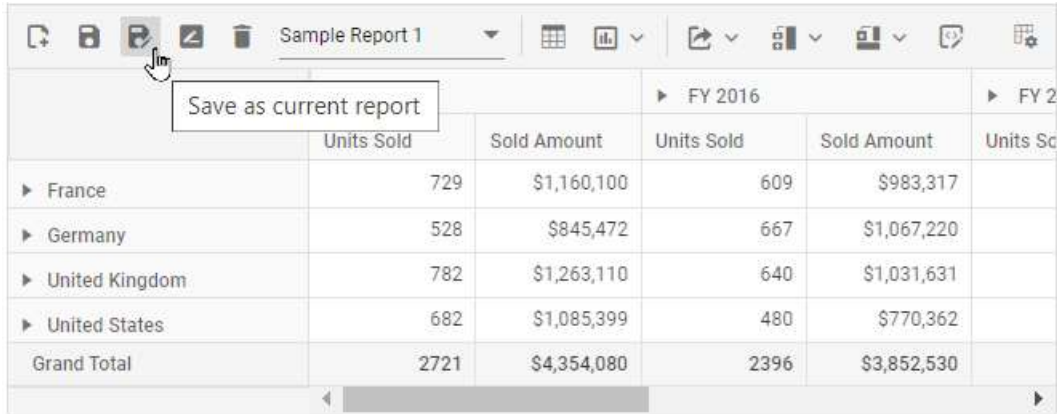
sqlConn.Close();
}
private SqlConnection OpenConnection()
{
// Replace with your own connection string.
string connectionString = @"<Enter your valid connection string here>";
SqlConnection sqlConn = new SqlConnection(connectionString);
sqlConn.Open();
return sqlConn;
}
private DataTable GetDataTable(SqlConnection sqlConn)
{
string xquery = "SELECT * FROM ReportTable";
SqlCommand cmd = new SqlCommand(xquery, sqlConn);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
return dt;
}
}
}
,

```



	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	729	\$1,160,100	609	\$983,317	
Germany	528	\$845,472	667	\$1,067,220	
United Kingdom	782	\$1,263,110	640	\$1,031,631	
United States	682	\$1,085,399	480	\$770,362	
Grand Total	2721	\$4,354,080	2396	\$3,852,530	

In the meantime, you can save a duplicate of the current report to the SQL Server database with a different name by selecting **“Save as current report”** from the toolbar. The [saveReport](#) event will then be triggered with the new report name **“Sample Report 1”** and the current report. You can save them to the SQL Server database after passing them to the Web API service, as mentioned above.



	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	729	\$1,160,100	609	\$983,317	
Germany	528	\$845,472	667	\$1,067,220	
United Kingdom	782	\$1,263,110	640	\$1,031,631	
United States	682	\$1,085,399	480	\$770,362	
Grand Total	2721	\$4,354,080	2396	\$3,852,530	

Loading a report

When you select the dropdown menu item from the toolbar, the [loadReport](#) event is triggered. In this event, an AJAX request is made to the **LoadReport** method of the Web API controller, passing the name of the selected report. The method uses this information to search for the report in the SQL database, fetch it, and load it into the pivot table.

For example, if the report name **"Sample Report 1"** is selected from a dropdown menu and passed, the **LoadReport** method will use that name to search for the report in the SQL database, retrieve it, and then load it into the pivot table.

[index.js]

`ts

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    columns: [
      { name: "Year", caption: "Production Year" },
      { name: "Quarter" },
    ],
    dataSource: getPivotData(),
    expandAll: false,
    filters: [],
    formatSettings: [{ name: "Amount", format: "C0" }],
    rows: [{ name: "Country" }, { name: "Products" }],
    values: [
      { name: "Sold", caption: "Units Sold" },
      { name: "Amount", caption: "Sold Amount" },
    ],
  },
  height: 350,
```

```
loadReport: function (args) {
fetch("https://localhost:44313/Pivot/LoadReport", {
method: "POST",
headers: {
Accept: "application/json",
"Content-Type": "application/json",
},
body: JSON.stringify({ reportName: args.reportName }),
})
.then(function (res) {
return res.json();
})
.then(function (response) {
if (response) {
var report = JSON.parse(response);
report.dataSourceSettings.dataSource =
pivotTableObj.dataSourceSettings.dataSource;
pivotTableObj.dataSourceSettings = report.dataSourceSettings;
}
});
},
toolbar: ['New', 'Save', 'SaveAs', 'Rename', 'Remove', 'Load',
'Grid', 'Chart', 'Export', 'SubTotal', 'GrandTotal', 'ConditionalFormatting', 'FieldList'],
allowExcelExport: true,
allowConditionalFormatting: true,
allowPdfExport: true,
showToolbar: true,
allowCalculatedField: true,
displayOption: { view: "Both" },
showFieldList: true,
height: "700",
});
pivotTableObj.appendTo("#PivotView");
```



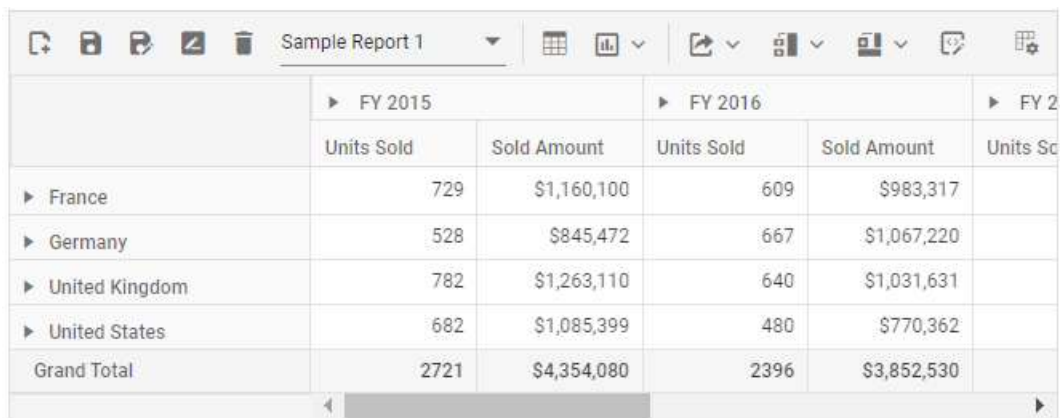
```
、  
  
[PivotController.cs]  
`c#  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.Data.SqlClient;  
using System.Data;  
namespace MyWebApp.Controllers  
{  
[ApiController]  
[Route("[controller]")]  
public class PivotController : ControllerBase  
{  
[HttpPost]  
[Route("Pivot/LoadReport")]  
public string LoadReport(Dictionary<string, string> args)  
{  
return LoadReportFromDB(args["ReportName"]);  
}  
public string LoadReportFromDB(string reportName)  
{  
SqlConnection sqlConn = OpenConnection();  
string report = string.Empty;  
foreach (DataRow row in GetDataTable(sqlConn).Rows)  
{  
if ((row["ReportName"] as string).Equals(reportName))  
{  
report = (string)row["Report"];  
break;  
}  
}  
sqlConn.Close();  
return report;  
}
```

```

private SqlConnection OpenConnection()
{
    // Replace with your own connection string.
    string connectionString = @"<Enter your valid connection string here>";
    SqlConnection sqlConn = new SqlConnection(connectionString);
    sqlConn.Open();
    return sqlConn;
}

private DataTable GetDataTable(SqlConnection sqlConn)
{
    string xquery = "SELECT * FROM ReportTable";
    SqlCommand cmd = new SqlCommand(xquery, sqlConn);
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    da.Fill(dt);
    return dt;
}
}
}
,

```



	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	729	\$1,160,100	609	\$983,317	
Germany	528	\$845,472	667	\$1,067,220	
United Kingdom	782	\$1,263,110	640	\$1,031,631	
United States	682	\$1,085,399	480	\$770,362	
Grand Total	2721	\$4,354,080	2396	\$3,852,530	

Renaming a report

When you select the **“Rename a current report”** option from the toolbar, the [renameReport](#) event is triggered. In this event, an AJAX request is made to the **RenameReport** method of the Web API controller, passing the current and new report names, where you can use the current report name to identify the report and resave it with the new report name in the SQL database.

For example, if we rename the current report from “**Sample Report 1**” to “**Sample Report 2**”, both “**Sample Report 1**” and “**Sample Report 2**” will be passed to the **RenameReport** method, which will rename the current report with the new report name “**Sample Report 2**” in the SQL database.

[index.js]

```
`ts
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    columns: [
      { name: "Year", caption: "Production Year" },
      { name: "Quarter" },
    ],
    dataSource: getPivotData(),
    expandAll: false,
    filters: [],
    formatSettings: [{ name: "Amount", format: "C0" }],
    rows: [{ name: "Country" }, { name: "Products" }],
    values: [
      { name: "Sold", caption: "Units Sold" },
      { name: "Amount", caption: "Sold Amount" },
    ],
  },
  height: 350,
  renameReport: function (args) {
    fetch("https://localhost:44313/Pivot/RenameReport", {
      method: "POST",
      headers: {
        Accept: "application/json",
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        reportName: args.reportName,
        renameReport: args.rename,
        isReportExists: args.isReportExists
      })
    },
```

```

    })
    .then(function (response) {
    pivotTableObj.fetchReport();
    });
    },
    toolbar: ['New', 'Save', 'SaveAs', 'Rename', 'Remove', 'Load',
    'Grid', 'Chart', 'Export', 'SubTotal', 'GrandTotal', 'ConditionalFormatting', 'FieldList'],
    allowExcelExport: true,
    allowConditionalFormatting: true,
    allowPdfExport: true,
    showToolbar: true,
    allowCalculatedField: true,
    displayOption: { view: "Both" },
    showFieldList: true,
    height: "700",
    });
    pivotTableObj.appendTo("#PivotView");
    、

```

[PivotController.cs]

```

`c#
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using System.Data;
namespace MyWebApp.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class PivotController : ControllerBase
    {
        [HttpPost]
        [Route("Pivot/RenameReport")]
        public void RenameReport(Dictionary<string, string, bool> args)
        {

```

```
RenameReportInDB(args["ReportName"], args["Rename"], args["isReportExists"]);
}
public void RenameReportInDB(string reportName, string renameReport, bool isReportExists)
{
    SqlConnection sqlConn = OpenConnection();
    SqlCommand cmd1 = null;
    if (isReportExists)
    {
        RemoveReportFromDB(renameReport);
    }
    foreach (DataRow row in GetDataTable(sqlConn).Rows)
    {
        if ((row["ReportName"] as string).Equals(reportName))
        {
            cmd1 = new SqlCommand("UPDATE ReportTable set ReportName=@RenameReport where ReportName
            like '%" + reportName + "%'", sqlConn);
            break;
        }
    }
    cmd1.Parameters.AddWithValue("@RenameReport", renameReport);
    cmd1.ExecuteNonQuery();
    sqlConn.Close();
}
private SqlConnection OpenConnection()
{
    // Replace with your own connection string.
    string connectionString = @"<Enter your valid connection string here>";
    SqlConnection sqlConn = new SqlConnection(connectionString);
    sqlConn.Open();
    return sqlConn;
}
private DataTable GetDataTable(SqlConnection sqlConn)
{

```

```

string xquery = "SELECT * FROM ReportTable";
SqlCommand cmd = new SqlCommand(xquery, sqlConn);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
return dt;
}
}
}
,

```

	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	729	\$1,160,100	609	\$983,317	
Germany	528	\$845,472	667	\$1,067,220	
United Kingdom	782	\$1,263,110	640	\$1,031,631	
United States	682	\$1,085,399	480	\$770,362	
Grand Total	2721	\$4,354,080	2396	\$3,852,530	

Deleting a report

When you select the **“Delete a current report”** option from the toolbar, the [removeReport](#) event is triggered. In this event, an AJAX request is made to the **RemoveReport** method of the Web API controller, passing the current report name to identify and delete the appropriate report from the SQL database.

* If the current report **n** from the pivot table is deleted, the pivot table will automatically load the last report from the report list.

* When a report is removed from a pivot table with only one report, the SQL database refreshes; however, the pivot table will continue to show the removed report until a new report is added to the pivot table.

For example, if we delete the current report **“Sample Report 2”** from the pivot table, the current report name **“Sample Report 2”** is passed to the **RemoveReport** method, which allows you to identify and delete the report from the SQL database.

[index.js]

`ts

```

var pivotTableObj = new ej.pivotview.PivotView({
dataSourceSettings: {

```

```
columns: [
{ name: "Year", caption: "Production Year" },
{ name: "Quarter" },
],
dataSource: getPivotData(),
expandAll: false,
filters: [],
formatSettings: [{ name: "Amount", format: "C0" }],
rows: [{ name: "Country" }, { name: "Products" }],
values: [
{ name: "Sold", caption: "Units Sold" },
{ name: "Amount", caption: "Sold Amount" },
],
},
height: 350,
removeReport: function (args) {
fetch("https://localhost:44313/Pivot/RemoveReport", {
method: "POST",
headers: {
Accept: "application/json",
"Content-Type": "application/json",
},
body: JSON.stringify({ reportName: args.reportName })
})
.then(function (response) {
pivotTableObj.fetchReport();
});
},
toolbar: ['New', 'Save', 'SaveAs', 'Rename', 'Remove', 'Load',
'Grid', 'Chart', 'Export', 'SubTotal', 'GrandTotal', 'ConditionalFormatting', 'FieldList'],
allowExcelExport: true,
allowConditionalFormatting: true,
allowPdfExport: true,
```

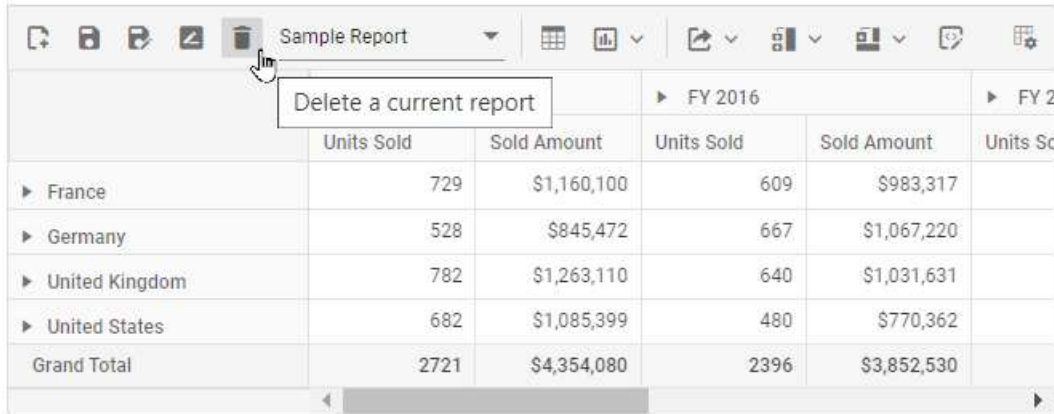
```
showToolBar: true,  
allowCalculatedField: true,  
displayOption: { view: "Both" },  
showFieldList: true,  
height: "700",  
});  
pivotTableObj.appendTo("#PivotView");  
`
```

```
[PivotController.cs]
```

```
`c#  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.Data.SqlClient;  
using System.Data;  
namespace MyWebApp.Controllers  
{  
    [ApiController]  
    [Route("[controller]")]  
    public class PivotController : ControllerBase  
    {  
        [HttpPost]  
        [Route("Pivot/RemoveReport")]  
        public void RemoveReport(Dictionary<string, string> args)  
        {  
            RemoveReportFromDB(args["ReportName"]);  
        }  
        public void RemoveReportFromDB(string reportName)  
        {  
            SqlConnection sqlConn = OpenConnection();  
            SqlCommand cmd1 = null;  
            foreach (DataRow row in GetDataTable(sqlConn).Rows)  
            {  
                if ((row["ReportName"] as string).Equals(reportName))  
                {
```



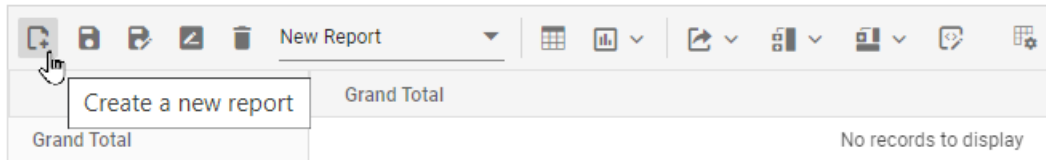
```
cmd1 = new SqlCommand("DELETE FROM ReportTable WHERE ReportName LIKE '%" + reportName +
"%'", sqlConn);
break;
}
}
cmd1.ExecuteNonQuery();
sqlConn.Close();
}
private SqlConnection OpenConnection()
{
// Replace with your own connection string.
string connectionString = @"<Enter your valid connection string here>";
SqlConnection sqlConn = new SqlConnection(connectionString);
sqlConn.Open();
return sqlConn;
}
private DataTable GetDataTable(SqlConnection sqlConn)
{
string xquery = "SELECT * FROM ReportTable";
SqlCommand cmd = new SqlCommand(xquery, sqlConn);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
return dt;
}
}
}
}
```



	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	729	\$1,160,100	609	\$983,317	
Germany	528	\$845,472	667	\$1,067,220	
United Kingdom	782	\$1,263,110	640	\$1,031,631	
United States	682	\$1,085,399	480	\$770,362	
Grand Total	2721	\$4,354,080	2396	\$3,852,530	

Adding a report

When you select the “**Create a new report**” option from the toolbar, the [newReport](#) event is triggered, followed by the [saveReport](#) event. To save this new report to the SQL database, use the [saveReport](#) event triggered later, and then follow the save report briefing in the preceding [topic](#).



	Grand Total
Grand Total	

No records to display

Limitations with respect to report manipulation

Below points need to be considered when saving the report to SQL Server database.

- **Data source:** Both raw data and aggregated data won't be saved and loaded from the database.
- **Conditional formatting:** The appearance of the pivot table, such as background color, font color, font family, and font size based on the specific conditions won't be saved and loaded from the database.
- **Hyperlinks:** Option to link external facts via pivot table cells won't be saved and loaded from the database.
- The pivot table should always load reports from the SQL database based on the data source that is currently bound to it.

In [this](#) GitHub repository, you can find our Javascript Pivot Table sample and ASP.NET Core Web Application to save and load reports from SQL Server database.

Events

FetchReport

The event [fetchReport](#) is triggered when dropdown list is clicked in the toolbar in-order to retrieve and populate saved reports. It has following parameter - **ReportName**. This event allows user to fetch the report names from local storage and populate the dropdown list.

LoadReport

The event [loadReport](#) is triggered when a report is selected from the dropdown list in the toolbar. It has following parameters - **Report** and **ReportName**. This event allows user to load the selected report to the pivot table.

NewReport

The event [newReport](#) is triggered when the new report icon is clicked in the toolbar. It has following parameter - **Report**. This event allows user to create new report and add to the report list.

RenameReport

The event [renameReport](#) is triggered when rename report icon is clicked in the toolbar. It has following parameters - **rename**, **report** and **reportName**. This event allows user to rename the selected report from the report list.

RemoveReport

The event [removeReport](#) is triggered when remove report icon is clicked in the toolbar. It has following parameters - **Report** and **ReportName**. This event allows user to remove the selected report from the report list.

SaveReport

The event [saveReport](#) is triggered when save report icon is clicked in the toolbar. It has following parameters - **Report** and **ReportName**. This event allows user to save the altered report to the report list.

ToolbarRender

The [toolbarRender](#) event is triggered when the toolbar is rendered. It has the **customToolbar** parameter. This event helps to customize the built-in toolbar items and to [include new toolbar item\(s\)](#).

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    sortSettings: [{ name: 'Country', order: 'Descending' }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: []
  },
  height: 350,
  toolbarRender: function (args) {
    args.customToolbar.splice(2, 0, {
      prefixIcon: 'e-pivot-format-toolbar e-icons', tooltipText:
'Custom Button',
      click: this.customButton.bind(this),
    });
    args.customToolbar.splice(3, 0, {
      prefixIcon: 'e-tool-expand e-icons', tooltipText:
'Expand/Collapse',
      click: this.toolbarClicked.bind(this),
    });
    args.customToolbar[0].align = "Left";
    args.customToolbar[1].align = "Center";
    args.customToolbar[2].align = "Right";
  },
  customButton: function(args) {
```

```

        // Here you can customize the click event for custom button
    },
    toolbarClicked: function(args) {
        pivotTableObj.dataSourceSettings.expandAll =
!pivotTableObj.dataSourceSettings.expandAll;
    },
    displayOption: { view: 'Both' },
    chartSettings: {
        value: 'Amount', enableExport: true, chartSeries: { type: 'Column',
animation: { enable: false } }, enableMultipleAxis: false,
    },
    toolbar: ['Save', 'Export', 'FieldList'],
    showToolbar: true,
    showFieldList: true,
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ActionBegin

The event [actionBegin](#) triggers when the UI actions such as switching between pivot table and pivot chart, changing chart types, conditional formatting, exporting, etc. that are present in toolbar UI begin. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action began. The following are the UI actions and their names:

Action	Action Name
-----	-----
New report	Add new report
Save report	Save current report
Save as report	Save as current report
Rename report	Rename current report
Remove report	Remove current report
Report change	Report change
Conditional Formatting	Open conditional formatting dialog
Number Formatting	Open number formatting dialog
Export menu	PDF export, Excel export, CSV export
Show Fieldlist	Open field list
Show Table	Show table view
Chart menu	Show chart view

| Sub-totals menu | Hide sub-totals, Show row sub-totals, Show column sub-totals, Show sub-totals |

| Grand totals menu | Hide grand totals, Show row grand totals, Show column grand totals, Show grand totals |

- **cancel**: It allows user to restrict the current action.

In the below sample, toolbar UI actions such as add new report and save current report can be restricted by setting the **args.cancel** option to **true** in the **actionBegin** event.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  showFieldList: true,
  displayOption: { view: 'Both' },
  chartSettings: {
    value: 'Amount', enableExport: true, chartSeries: { type: 'Column',
animation: { enable: false } }, enableMultipleAxis: false,
  },
  toolbar: ['New', 'Save', 'SaveAs', 'Rename', 'Remove', 'Load',
'Grid', 'Chart', 'MDX', 'Export', 'SubTotal', 'GrandTotal',
'ConditionalFormatting', 'FieldList'],
  allowExcelExport: true,
  allowConditionalFormatting: true,
  allowPdfExport: true,
  showToolbar: true,
  actionBegin: function (args) {
    if (args.actionName == 'Add new report' || args.actionName == 'Save
current report') {
      args.cancel = true;
    }
  },
  height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ActionComplete

The event [actionComplete](#) triggers when the UI actions such as switching between pivot table and pivot chart, changing chart types, conditional formatting, exporting, etc. that are present in toolbar UI, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. The following are the UI actions and their names:

Action	Action Name
-----	-----
New report	New report added
Save report	Report saved
Save as report	Report re-saved
Rename report	Report renamed
Remove report	Report removed
Report change	Report changed
Conditional Formatting	Conditionally formatted
Number Formatting	Number formatted
Export menu	PDF exported, Excel exported, CSV exported
Show Fieldlist	Field list closed
Show Table	Table view shown
Sub-totals menu	Sub-totals hidden, Row sub-totals shown, Column sub-totals shown, Sub-totals shown
Grand totals menu	Grand totals hidden, Row grand totals shown, Column grand totals shown, Grand totals shown

- **actionInfo**: It holds the unique information about the current UI action. For example, while adding new report, the event argument contains information such as report name and the action name.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  showFieldList: true,
  displayOption: { view: 'Both' },
  chartSettings: {
    value: 'Amount', enableExport: true, chartSeries: { type: 'Column',
animation: { enable: false } }, enableMultipleAxis: false,
  },
  toolbar: ['New', 'Save', 'SaveAs', 'Rename', 'Remove', 'Load',
'Grid', 'Chart', 'MDX', 'Export', 'SubTotal', 'GrandTotal',
'ConditionalFormatting', 'FieldList'],
  allowExcelExport: true,
```



```

        allowConditionalFormatting: true,
        allowPdfExport: true,
        showToolBar: true,
        actionComplete: function (args) {
            if (args.actionName == 'New report added' || args.actionName ==
'Report saved') {
                // Triggers when the toolbar UI actions such as add new report
and save current report icon are completed.
            }
        },
        height: 350
    });
    pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div class="container">
    <div id="PivotTable"></div>
  </div>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ActionFailure

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- **actionName**: It holds the name of the current action failed. The following are the UI actions and their names:

Action	Action Name
-----	-----
New report	Add new report
Save report	Save current report
Save as report	Save as current report
Rename report	Rename current report
Remove report	Remove current report
Report change	Report change
Conditional Formatting	Open conditional formatting dialog
Number Formatting	Open number formatting dialog
Export menu	PDF export, Excel export, CSV export
Show Fieldlist	Open field list
Show Table	Show table view
Chart menu	Show chart view
Sub-totals menu	Hide sub-totals, Show row sub-totals, Show column sub-totals, Show sub-totals
Grand totals menu	Hide grand totals, Show row grand totals, Show column grand totals, Show grand totals

- **errorInfo**: It holds the error information of the current UI action.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,

```

```

        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    showGroupingBar: true,
    showFieldList: true,
    allowCalculatedField: true,
    showToolBar: true,
    displayOption: { view: 'Both' },
    toolbar: ['New', 'Save', 'Rename', 'Remove', 'Load',
'Grid', 'Chart', 'MDX', 'Export', 'SubTotal', 'GrandTotal',
'ConditionalFormatting', 'FieldList'],
    allowExcelExport: true,
    allowConditionalFormatting: true,
    allowPdfExport: true,
    actionFailure: function (args) {
        if (args.actionName == 'Add new report' || args.actionName == 'Save
current report') {
            // Triggers when the current UI action fails to achieve the
desired result.
        }
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Toolbar Component](#)
- [Excel Exporting](#)
- [PDF Exporting](#)

Tool tip in EJ2 JavaScript Pivotview control

The tooltip can be enabled or disabled by setting the [showTooltip](#) property to **true**. By default, tooltip is enabled in the pivot table.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    showTooltip: false,
    height: 350

```

```
});  
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 Pivot Grid</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="Typescript Pivot Grid Control">  
  <meta name="author" content="Syncfusion">  
  <link href="index.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
base/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
buttons/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
calendars/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
dropdowns/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
grids/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
charts/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
inputs/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
lists/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
navigations/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
popups/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-  
pivotview/styles/material.css" rel="stylesheet">  
  
  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-  
awesome.min.css" rel="stylesheet">  
  
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"  
type="text/javascript"></script>  
  <script src="es5-datasource.js" type="text/javascript"></script>  
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type  
="text/javascript"></script>  
</head>  
<body>  
  
  <div id="container">  
    <div>  
      <div id="PivotTable"></div>  
    </div>  
  </div>  
<script>  
var ele = document.getElementById('container');  
if(ele) {  
  ele.style.visibility = "visible";  
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip Template

User can design their own tooltip by setting the property [tooltipTemplate](#) with own HTML elements. The property accepts both HTML string and ID attribute. The following place holders are available to display its dynamic values inside the HTML elements.

`\${rowHeaders}` – Row headers of the selected value cell.

`\${columnHeaders}` – Column headers of the selected value cell.

`\${rowFields}` – Row fields of the selected value cell.

`\${columnFields}` – Column fields of the selected value cell.

`\${valueField}` – Field name of the selected value cell.

`\${aggregateType}` – Aggregate type of the selected value cell.

`\${value}` - Formatted value of the selected value cell.

The tooltip customization is common for both pivot table and pivot chart or it can be done individually as well. To customize the pivot table tooltip, the above procedure needs to be followed. To customize the pivot chart tooltip alone use **template** property of tooltip under [chartSettings](#).

In the below sample, the pivot table and pivot chart shows customized tooltip layouts.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    filters: []
  },
  height: 350,
  displayOption: { view: 'Both' },
  chartSettings: {
    value: 'Amount', chartSeries: { type: 'Column', animation: {
enable: false } },
    tooltip: { template: '<span class="wrap">`${aggregateType}` of
`${valueField}`: `${value}</span>' }
  },
  toolbar: ['Grid', 'Chart'],
  showToolbar: true,
  tooltipTemplate: "#Template"
});
pivotTableObj.appendTo("#PivotTable");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="PivotTable"></div>
  </div>
  <script id="Template" type="text/x-template">
    <div class='wrap'>
      <div>
        <span class='pivotTooltipHeader'>Row Headers :</span>
      </div>
      <div>
        <span class='pivotTooltipValue'>${columnHeaders}${rowHeaders}</span>
      </div>
      <div>
        <span class='pivotTooltipHeader'>Row Fields :
      </span>
        <span class='pivotTooltipValue'>${rowFields}</span>
      </div>
      <div>
        <span class='pivotTooltipHeader'>Column Headers
      </span>
    </div>
  </script>

```

```

        <span
class='pivotTooltipValue'>${columnHeaders}</span>
    </div>
    <div>
        <span class='pivotTooltipHeader'>Column Fields
: </span>
        <span
class='pivotTooltipValue'>${columnFields}</span>
    </div>
    <div>
        <span class='pivotTooltipHeader'>Value Field
: </span>
        <span class='pivotTooltipValue'>${valueField}</span>
    </div>
    <div>
        <span class='pivotTooltipHeader'>Value : </span>
        <span class='pivotTooltipValue'>${value}</span>
    </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Css customization in EJ2 JavaScript Pivotview control

Hiding Axis

The visibility of row, column, value and filter axis in Field List and Grouping Bar can be changed using custom CSS setting. To do so, please refer the code sample below:

```
PivotTable .e-group-columns {
```

```
display: none;
```

```
}
```

```
PivotTable .e-group-filters {
```

```
height: 71px !important;
```

```
}
```

```
PivotTable_PivotFieldList_Wrapper .e-field-list-columns{
```

```
display: none;
```

```
}
```

```
PivotTable_PivotFieldList_Wrapper .e-field-list-values{
```

```
margin-top: 0px;
```

```
height: 338px;
```



```

}
,

```

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 320,
  showGroupingBar: true,
  showFieldList: true
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Text Alignment

The alignment of text inside row headers, column headers, value cells and summary cells can be changed using custom CSS setting. To do so, please refer the code sample below:

```

.e-pivotview .e-valuescontent {
text-align: center !important;
}

```

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    showGroupingBar: true,
    showFieldList: true
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Pivot Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize header, value and summary cell style

The elements in pivot table like header cell, value cell and summary cell style can be customized using built-in CSS names. To do so, please refer the code sample below:

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 320,
  showGroupingBar: true,
  showFieldList: true
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="PivotTable"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Printing and Exporting

Print

The rendered pivot table can be printed directly from the browser by invoking the [print](#) method from the grid's instance. The below sample code illustrates the print option being invoked by an external button click.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [
            { name: 'Year', caption: 'Production Year' },
            { name: 'Quarter' },
        ],
        values: [
            { name: 'Sold', caption: 'Units Sold' },
            { name: 'Amount', caption: 'Sold Amount' },
        ],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ content: 'Print', cssClass: `e-
primary`, isPrimary: true });
exportBtn.appendTo('#print');
document.getElementById('print').onclick = function () {
    // Method used to print the pivot table.
    pivotTableObj.grid.print();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Pivot Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div> <button id='print' style="margin-bottom: 5px;"></button></div>
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Similarly, to print the pivot chart, use the [print](#) method from the chart's instance. The below sample code illustrates the print option being invoked by an external button click.

To use pivot chart, you need to inject the `PivotChart` module in the pivot table.

To display the pivot chart, set the [displayOption](#) property to either **Chart** or **Both**.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [
      { name: 'Year', caption: 'Production Year' },
      { name: 'Quarter' },
    ],
    values: [
      { name: 'Sold', caption: 'Units Sold' },
      { name: 'Amount', caption: 'Sold Amount' },
    ],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: [],
  },
  height: 350,
  displayOption: { view: 'Chart' }
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ content: 'Print', cssClass: `e-
primary`, isPrimary: true });
exportBtn.appendTo('#print');
document.getElementById('print').onclick = function () {
  // Method used to print the pivot chart.
  pivotTableObj.chart.print();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div> <button id='print' style="margin-bottom: 5px;"></button></div>
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Excel export in EJ2 JavaScript Pivotview control

The Excel export allows pivot table data to export as Excel document. To enable Excel export in the pivot table, set the `allowExcelExport` as **true**. You need to use the `excelExport` method for Excel exporting.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],

```



```

        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    allowExcelExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#excel');
document.getElementById('excel').onclick = function () {
    pivotTableObj.excelExport();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="excel" type="button" value="Export To Excel" name="Export
To Excel">
        <div id="PivotTable"></div>
    </div>

```

```

        <br>
        <div id="PivotTable2"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple pivot table exporting

The Excel export provides an option to export multiple pivot table data in the same Excel file.

Same WorkSheet

The Excel export provides support to export multiple pivot tables in same sheet. To export in same sheet, define `multipleExport.type` as `AppendToSheet` in `ExcelExportProperties`. It has an option to provide blank rows between pivot tables and these blank row(s) count can be defined using the `multipleExport.blankRows` property.

By default, `multipleExport.blankRows` value is 5 between pivot tables within the same sheet.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 300,
    allowExcelExport: true
});
var pivotTableObj2 = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        rows: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        columns: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 300,
    allowExcelExport: true
});
pivotTableObj.appendTo('#PivotTable');

```

```

pivotTableObj2.appendTo('#PivotTable2');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#excel');
document.getElementById('excel').onclick = function () {
    var excelExportProperties = {
        multipleExport: { type: 'AppendToSheet', blankRows: 2 }
    };
    var firstGridExport =
pivotTableObj.grid.excelExport(excelExportProperties, true);
    firstGridExport.then(function (fData) {
        pivotTableObj2.excelExport(excelExportProperties, false, fData);
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="excel" type="button" value="Export To Excel" name="Export
To Excel">

```

```

        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

New WorkSheet

Excel export provides support to export multiple pivot tables into new sheets. To export in new sheets, define `multipleExport.type` as `NewSheet` in `ExcelExportProperties`.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 300,
    allowExcelExport: true
});
var pivotTableObj2 = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        rows: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        columns: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 300,
    allowExcelExport: true
});
pivotTableObj.appendTo('#PivotTable');
pivotTableObj2.appendTo('#PivotTable2');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#excel');
document.getElementById('excel').onclick = function () {
    var excelExportProperties = {
        multipleExport: { type: 'NewSheet' }
    }
}

```

```

    };
    var firstGridExport =
    pivotTableObj.grid.excelExport(excelExportProperties, true);
    firstGridExport.then(function (fData) {
        pivotTableObj2.excelExport(excelExportProperties, false, fData);
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="excel" type="button" value="Export To Excel" name="Export
To Excel">
    <div id="PivotTable"></div>
    <br>
    <div id="PivotTable2"></div>
  </div>
  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing the pivot table style while exporting

The Excel export provides an option to change colors for headers, caption and records in pivot table before exporting. In-order to apply colors, define **theme** settings in **excelExportProperties** object and pass it as a parameter to the [excelExport](#) method.

By default, material theme is applied to exported Excel document.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    allowExcelExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#excel');
document.getElementById('excel').onclick = function () {
    var excelExportProperties = {
        theme: {
            header: { fontName: 'Segoe UI', fontColor: '#666666' },
            record: { fontName: 'Segoe UI', fontColor: '#666666' },
            caption: { fontName: 'Segoe UI', fontColor: '#666666' }
        }
    };
    pivotTableObj.excelExport(excelExportProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="excel" type="button" value="Export To Excel" name="Export
To Excel">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add header and footer while exporting

The Excel export provides an option to include header and footer content for the excel document before exporting. In-order to add header and footer, define **header** and **footer** properties in **excelExportProperties** object and pass it as a parameter to the [excelExport](#) method.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,

```

```

        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    allowExcelExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#excel');
document.getElementById('excel').onclick = function () {
    var excelExportProperties = {
        header: {
            headerRows: 2,
            rows: [
                { cells: [{ colSpan: 4, value: "Pivot Grid", style: {
fontColor: '#C67878', fontSize: 20, hAlign: 'Center', bold: true, underline:
true } } ] }
            ],
        },
        footer: {
            footerRows: 4,
            rows: [
                { cells: [{ colSpan: 4, value: "Thank you for your
business!", style: { hAlign: 'Center', bold: true } } ] },
                { cells: [{ colSpan: 4, value: "!Visit Again!", style: {
hAlign: 'Center', bold: true } } ] }
            ]
        }
    };
    pivotTableObj.excelExport(excelExportProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="excel" type="button" value="Export To Excel" name="Export
To Excel">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing the file name while exporting

The Excel export provides an option to change file name of the document before exporting. In-order to change the file name, define **fileName** property in **excelExportProperties** object and pass it as a parameter to the [excelExport](#) method.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,

```

```

        allowExcelExport: true
    });
    pivotTableObj.appendTo('#PivotTable');
    var exportBtn = new ej.buttons.Button({ isPrimary: true });
    exportBtn.appendTo('#excel');
    document.getElementById('excel').onclick = function () {
        var excelExportProperties = {
            fileName: 'sample.xlsx'
        };
        pivotTableObj.excelExport(excelExportProperties);
    };

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="excel" type="button" value="Export To Excel" name="Export
To Excel">
        <div id="PivotTable"></div>
    <br>

```

```

        <div id="PivotTable2"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

CSV Export

Also, the Excel export allows pivot table data to be exported in CSV file format. To export pivot table in CSV file format, you need to use the `csvExport` method.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    allowExcelExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#excel');
document.getElementById('excel').onclick = function () {
    pivotTableObj.csvExport();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="excel" type="button" value="Export To Excel" name="Export
To Excel">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Virtual Scroll Data

You can export the pivot table virtual scroll data as Excel/CSV document by using PivotEngine export without any performance degradation. To enable PivotEngine export in the pivot table, set the `allowExcelExport` as true. You need to use the `exportToExcel` method for PivotEngine export.

To use PivotEngine export, You need to inject the `ExcelExport` module in pivot table.

PivotEngine export will be performed while enabling virtual scrolling by default.

Virtual Scroll Data Excel Export

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,

```

```

        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    allowExcelExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#excel');
document.getElementById('excel').onclick = function () {
    pivotTableObj.excelExportModule.exportToExcel('Excel');
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <input id="excel" type="button" value="Export To Excel" name="Export
To Excel">
      <div id="PivotTable"></div>
      <br>
      <div id="PivotTable2"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Virtual Scroll Data CSV Export

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 320,
  allowExcelExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#excel');
document.getElementById('excel').onclick = function () {
  var excelExportProperties = {
    fileName: 'csvexport.csv',
  };
  pivotTableObj.csvExport(excelExportProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="excel" type="button" value="Export To Excel" name="Export
To Excel">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export all pages

The pivot engine exports the entire virtual data of the pivot table (i.e. the data that contains all of the records used to render the complete pivot table) as an Excel/CSV document. To export just the current viewport of the pivot table, set the [exportAllPages](#) property to **false**. To use the pivot engine export, add the **ExcelExport** module into the pivot table.

By default, the pivot engine export will be performed while virtual scrolling is enabled.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    rows: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    values: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 320,
  allowExcelExport: true,
  exportAllPages: false,
  enableVirtualization: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#excel');
document.getElementById('excel').onclick = function () {
  pivotTableObj.excelExport();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="excel" type="button" value="Export To Excel" name="Export
To Excel">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Events

ExcelQueryCellInfo

The event `excelQueryCellInfo` triggers while framing each row and value cell during Excel export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- **value** - It holds the cell value.
- **column** - It holds column information for the current cell.
- **data** - It holds the entire row data across the current cell.
- **style** - It holds the style properties for the cell.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    gridSettings: {
        columnWidth: 120,

```

```

        excelQueryCellInfo: function (args) {
            // triggers every time for header cell while rendering
        },
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ExcelHeaderQueryCellInfo

The event `excelHeaderQueryCellInfo` triggers on framing each header cell during Excel export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- `cell` - It holds the current cell information.
- `style` - It holds the style properties for the cell.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    gridSettings: {
        columnWidth: 120,
        excelHeaderQueryCellInfo: function (args) {
            // triggers every time for header cell while rendering
        },
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ExportComplete

The event [exportComplete](#) is triggered after the pivot table data has been exported to an Excel/CSV document. You can use this event to acquire blob stream data for further customization and processing at your end by passing the `isBlob` parameter as **true** when using the [excelExport](#) method. It has the following parameters:

- **type** - It holds the current export type such as PDF, Excel, and CSV.
- **promise** - It holds the promise object for blob data.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    rows: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    values: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 320,
  allowExcelExport: true,
  enableVirtualization: true,
  exportComplete: function (args) {
    if (args.promise !== null)
      args.promise.then(function (e) {
        console.log(e.blobData);
      });
  }
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#excel');
document.getElementById('excel').onclick = function () {
  var excelExportProperties = {
    fileName: 'excelexport.xlsx',
  };
  pivotTableObj.excelExport(excelExportProperties, false, null, true);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="excel" type="button" value="Export To Excel" name="Export
To Excel">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

[See Also](#)

- [PDF Exporting](#)

Pdf export in EJ2 JavaScript Pivotview control

PDF export allows exporting pivot table data as PDF document. To enable PDF export in the pivot table, set the `allowPdfExport` as true. You need to use the `pdfExport` method for PDF exporting.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,

```

```

        allowPdfExport: true
    });
    pivotTableObj.appendTo('#PivotTable');
    var exportBtn = new ej.buttons.Button({ isPrimary: true });
    exportBtn.appendTo('#pdf');
    document.getElementById('pdf').onclick = function () {
        pivotTableObj.pdfExport();
    };

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
    <script>
    var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple pivot table exporting

PDF export provides an option for exporting multiple pivot tables to same file. In this exported document, each pivot table will be exported to new page of document in same file.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 300,
    allowPdfExport: true
});
var pivotTableObj2 = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        rows: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        columns: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 300,
    allowPdfExport: true
});
pivotTableObj.appendTo('#PivotTable');
pivotTableObj2.appendTo('#PivotTable2');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    var firstGridPdfExport = pivotTableObj.grid.pdfExport({}, true);
    firstGridPdfExport.then(function (pdfData) {
        pivotTableObj2.pdfExport({}, false, pdfData);
    });
};

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="pdf" type="button" value="PDF Export" name="PDF Export">
    <div id="PivotTable"></div>
    <br>
    <div id="PivotTable2"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Export table and chart into the same document

When the [displayOption](#) is set to **Both**, you can export both the table and the chart into the same PDF document. To achieve this, use the [pdfExport](#) method and set the `exportBothTableAndChart` parameter to **true**.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  allowPdfExport: true,
  displayOption: { view: 'Both', primary: 'Table' },
  chartSettings: {
    value: 'Amount', enableExport: true, chartSeries: { type: 'Column' }
  },
  height: 320,
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
  pivotTableObj.pdfExport(null, false, null, false, true);
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization during PDF export

PDF export provides option to customize mapping of pivot table to the exported PDF document.

To add header and footer while exporting

You can customize text, page number, line, page size and changing orientation in header and footer of the exported document.

To add a text in header/footer

You can add text either in header or footer of the exported PDF document like in the below code example.

```

`js
var pdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{

```

```

type: 'Text',
value: "Northwind Traders",
position: { x: 0, y: 50 },
style: { textBrushColor: '#000000', fontSize: 13 }
},
]
}
}
,

```

To draw a line in header/footer

You can add line either in header or footer of the exported PDF document like in the below code example.

Supported line styles:

- dash
- dot
- dashdot
- dashdotdot
- solid

```

`js
var pdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{
type: 'Line',
style: { penColor: '#000080', penSize: 2, dashStyle: 'Solid' },
points: { x1: 0, y1: 4, x2: 685, y2: 4 }
}
]
}
}
,

```

[Add page number in header/footer](#)

You can add page number either in header or footer of exported PDF document like in the below code example.

Supported page number types:

- LowerLatin - a, b, c,
- UpperLatin - A, B, C,
- LowerRoman - i, ii, iii,
- UpperRoman - I, II, III,
- Number - 1,2,3.

```
`js
var pdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'PageNumber',
        pageNumberType: 'Arabic',
        format: 'Page {$current} of {$total}', //optional
        position: { x: 0, y: 25 },
        style: { textBrushColor: '#ffff80', fontSize: 15, hAlign: 'Center' }
      }
    ]
  }
},
`
```

The below code illustrates the PDF export customization options.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
```

```

        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    allowPdfExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    var pdfExportProperties = {
        header: {
            fromTop: 0,
            height: 130,
            contents: [
                {
                    type: 'Text',
                    value: "Pivot Grid",
                    position: { x: 0, y: 50 },
                    style: { textBrushColor: '#000000', fontSize: 13,
dashStyle: 'Solid', hAlign: 'Center' }
                }
            ]
        },
        footer: {
            fromBottom: 160,
            height: 150,
            contents: [
                {
                    type: 'PageNumber',
                    pageNumberType: 'Arabic',
                    format: 'Page { $current } of { $total }',
                    position: { x: 0, y: 25 },
                    style: { textBrushColor: '#02007a', fontSize: 15 }
                }
            ]
        }
    };
    pivotTableObj.pdfExport(pdfExportProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add an image in header/footer

You can add image (Base64 string) either in header or footer of the exported PDF document like in the below code example.

```

`js
var pdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{
type: 'Image',

```

```

src: image,
position: { x: 20, y: 10 },
size: { height: 100, width: 100 },
}
]
}
}
,

```

The below code illustrates the PDF export customization options.

INDEX.TS

```

import { PivotView, IDataset } from '@syncfusion/ej2-pivotview';
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { Button } from '@syncfusion/ej2-buttons';
import { pivotData } from './datasource.ts';
import { image } from './image.ts';
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataset[],
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    allowPdfExport: true,
    height: 320
});
pivotTableObj.appendTo('#PivotTable');
let exportBtn: Button = new Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    let pdfExportProperties: PdfExportProperties = {
        header: {
            fromTop: 0,
            height: 130,
            contents: [
                {
                    type: 'Image',
                    src: image,
                    position: { x: 20, y: 10 },
                    size: { height: 100, width: 100 },
                }
            ]
        }
    };
};

```



```

    pivotTableObj.pdfExport(pdfExportProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="image.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <input id="pdf" type="button" value="PDF Export" name="PDF Export">
    <div id="PivotTable"></div>
    <br>
    <div id="PivotTable2"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if(ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing the file name while exporting

The PDF export provides an option to change file name of the document before exporting. In-order to change the file name, define **fileName** property in **pdfExportProperties** object and pass it as a parameter to the [pdfExport](#) method.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 320,
  allowPdfExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
  var pdfExportProperties = {
    fileName: 'sample.pdf'
  };
  pivotTableObj.pdfExport(pdfExportProperties);
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing page orientation while exporting

The PDF export provides an option to change page orientation of the document before exporting. In order to change the page orientation, define **pageOrientation** property in **pdfExportProperties** object and pass it as a parameter to the [pdfExport](#) method. By default, the page orientation will be in **Portrait** and it can be changed to **Landscape** based on user requirement.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    allowPdfExport: true
});
pivotTableObj.appendTo('#PivotTable');

```

```

var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    var pdfExportProperties = {
        pageOrientation: 'Landscape'
    };
    pivotTableObj.pdfExport(pdfExportProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
    <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing page size while exporting

The PDF export provides an option to change page size of the document before exporting. In-order to change the page size, define **pageSize** property in **pdfExportProperties** object and pass it as a parameter to the [pdfExport](#) method.

Supported page sizes are: Letter, Note, Legal, A0, A1, A2, A3, A5, A6, A7, A8, A9, B0, B1, B2, B3, B4, B5, ArchA, ArchB, ArchC, ArchD, Arche, Flsa, HalfLetter, Letter11x17, Ledger.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    allowPdfExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    var pdfExportProperties = {
        pageSize: 'Letter'
    };
    pivotTableObj.pdfExport(pdfExportProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing document width and height while exporting

Before exporting, you can change the height and width of the PDF document. To achieve this, use the **height** and **width** properties in the [beforeExport](#) event.

This option is only available if [enableVirtualization](#) is set to **true**. In addition, the [VirtualScroll](#) and [PDFExport](#) modules must be injected into the pivot table.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,

```

```

        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    allowPdfExport: true,
    enableVirtualization: true,
    height: 320,
    beforeExport: function (args) {
        args.width = pivotTableObj.element.offsetWidth;
        args.height = pivotTableObj.element.offsetHeight;
    },
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    pivotTableObj.pdfExport();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the table column count while exporting

Before exporting, you can split and export the pivot table columns on each page of the PDF document by using the **columnSize** property in the [beforeExport](#) event.

This option is only available if [enableVirtualization](#) is set to **true**. In addition, the **VirtualScroll** and **PDFExport** modules must be injected into the pivot table.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    allowPdfExport: true,
    enableVirtualization: true,
    height: 320,
    beforeExport: function (args) {
        args.columnSize = 6;
    },
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {

```



```

    pivotTableObj.pdfExport();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="pdf" type="button" value="PDF Export" name="PDF Export">
    <div id="PivotTable"></div>
    <br>
    <div id="PivotTable2"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing the table's column width and row height while exporting

You can change the column width and row height in the PDF document during the pivot table export by using the [onPdfCellRender](#) event. Within this event, the `args.column.width` property allows you to change the width of specific columns.

As shown in the code example below, the “Unit Sold” column under “FY 2015” is changed to a width of 60 pixels.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  allowPdfExport: true,
  enableVirtualization: true,
  height: 320,
  onPdfCellRender: function (args) {
    if (args.pivotCell && args.pivotCell.valueSort &&
args.pivotCell.valueSort.levelName === 'FY 2015.Units Sold') {
      args.column.width = 60
    }
  },
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
  pivotTableObj.pdfExport({}, false, null, false, true);
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Similarly, you can change the height of specific rows in the PDF document by using the `args.cell.height` property in the [onPdfCellRender](#) event.

As shown in the code example below, the “**Mountain Bikes**” row under “**France**” is changed to a height of **30** pixels.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],

```

```

        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    allowPdfExport: true,
    enableVirtualization: true,
    height: 320,
    onPdfCellRender: function (args) {
        if (args.pivotCell && args.pivotCell.valueSort &&
args.pivotCell.valueSort.levelName === 'France.Mountain Bikes') {
            args.cell.height = 30
        }
    },
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    pivotTableObj.pdfExport({}, false, null, false, true);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

This option is only available if [enableVirtualization](#) is set to **true**. In addition, the **VirtualScroll** and **PDFExport** modules must be injected into the pivot table.

Changing the pivot table style while exporting

The PDF export provides an option to change colors for headers, caption and records in pivot table before exporting. In-order to apply colors, define **theme** settings in **pdfExportProperties** object and pass it as a parameter to the [pdfExport](#) method.

By default, material theme is applied to exported PDF document.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name: 'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount', caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    allowPdfExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    var pdfExportProperties = {
        theme: {
            header: {

```

```

        fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true, borders: { color: '#64FA50', lineStyle: 'Thin' }
    },
    record: {
        fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true
    },
    caption: {
        fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true
    }
}
};
pivotTableObj.pdfExport(pdfExportProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD009 -->

Changing default font while exporting

By default, the pivot table uses "Helvetica" font in the exported document. But it can be changed using the [theme](#) property in [pdfExportProperties](#).

The available built-in fonts are,

- Helvetica
- TimesRoman
- Courier
- Symbol
- ZapfDingbats

```
`js
```

```

var pdfExportProperties = {
  theme: {
    header: { font: new ej.pdfexport.PdfStandardFont(ej.pdfexport.PdfFontFamily.TimesRoman, 11,
    ej.pdfexport.PdfFontStyle.Bold) },
    caption: { font: new ej.pdfexport.PdfStandardFont(ej.pdfexport.PdfFontFamily.TimesRoman, 9) },
    record: { font: new ej.pdfexport.PdfStandardFont(ej.pdfexport.PdfFontFamily.TimesRoman, 10) }
  }
};
`

```

Adding custom font while exporting

In addition to existing built-in fonts, custom fonts can also be used. The custom font should be in **Base64** format and mention it in **PdfTrueTypeFont** class. In the following example, we have used **Advent Pro** font family that supports **Hungarian** language.

INDEX.JS

```

var pivotTableObj = new ej.pivottview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,

```

```

        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    allowPdfExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    var pdfExportProperties = {
        theme: {
            header: { font: new
ej.pdfexport.PdfTrueTypeFont(window.base64AlgeriaFont, 11) },
            caption: { font: new
ej.pdfexport.PdfTrueTypeFont(window.base64AlgeriaFont, 9) },
            record: { font: new
ej.pdfexport.PdfTrueTypeFont(window.base64AlgeriaFont, 10) }
        }
    };
    pivotTableObj.pdfExport(pdfExportProperties);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The non-English alphabets can also be exported properly by setting its appropriate font.

Virtual Scroll Data

You can export the pivot table virtual scroll data as PDF document by using PivotEngine export without any performance degradation. To enable PivotEngine export in the pivot table, set the `allowPdfExport` as true. You need to use the `exportToPDF` method for PivotEngine export.

To use PivotEngine export, You need to inject the `PDFExport` module in pivot table.

PivotEngine export will be performed while enabling virtual scrolling by default

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 320,
    allowPdfExport: true

```

```
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    pivotTableObj.pdfExportModule.exportToPDF();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Repeat row headers

Repeat row headers on each page can be achieved using PivotEngine export option. To disable repeat row headers, you need to set `allowRepeatHeader` to **false** in `beforeExport` event. You need to use the `exportToPDF` method for PivotEngine export.

To use PivotEngine export, You need to inject the `PDFExport` module in pivot table.

By default, repeat row headers is enabled in the PivotEngine export.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  beforeExport: function(args) {
    args.allowRepeatHeader = false;
  },
  height: 320,
  allowPdfExport: true
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
  pivotTableObj.pdfExportModule.exportToPDF();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export all pages

The pivot engine exports the entire virtual data of the pivot table (i.e. the data that contains all of the records used to render the complete pivot table) as a PDF document. To export just the current viewport of the pivot table, set the [exportAllPages](#) property to **false**. To use the pivot engine export, add the **PDFExport** module into the pivot table.

By default, the pivot engine export will be performed while virtual scrolling is enabled.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],

```

```

        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        rows: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        values: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    allowPdfExport: true,
    exportAllPages: false,
    enableVirtualization: true,
    height: 320
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    pivotTableObj.pdfExport();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Events

PdfQueryCellInfo

The event `pdfQueryCellInfo` triggers on framing each row and value cell during PDF export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- **value** - It holds the cell value.
- **column** - It holds column information for the current cell.
- **data** - It holds the entire row data across the current cell.
- **style** - It holds the style properties for the cell.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    gridSettings: {
        columnWidth: 120,
        pdfQueryCellInfo: function (args) {
            // triggers every time for header cell while rendering
        },
    }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

PdfHeaderQueryCellInfo

The event `pdfHeaderQueryCellInfo` triggers on framing each column header cell during PDF export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- `cell` - It holds the current rendering cell information.
- `style` - It holds the style properties for the cell.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  gridSettings: {
    columnWidth: 120,
    pdfHeaderQueryCellInfo: function (args) {
      // triggers every time for header cell while rendering
    },
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

ExportComplete

The event [exportComplete](#) is triggered after the pivot table data has been exported to an PDF document. You can use this event to acquire blob stream data for further customization and processing at your end by passing the `isBlob` parameter as `true` when using the [pdfExport](#) method. It has the following parameters:

- `type` - It holds the current export type such as PDF, Excel, and CSV.
- `promise` - It holds the promise object for blob data.

INDEX.JS

```

var pivotTableObj = new ej.pivotview.PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],

```

```

        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        rows: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        values: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    allowPdfExport: true,
    enableVirtualization: true,
    height: 320,
    exportComplete: function (args) {
        if (args.promise !== null)
            args.promise.then(function (e) {
                console.log(e.blobData);
            });
    }
});
pivotTableObj.appendTo('#PivotTable');
var exportBtn = new ej.buttons.Button({ isPrimary: true });
exportBtn.appendTo('#pdf');
document.getElementById('pdf').onclick = function () {
    var pdfExportProperties = {
        fileName: 'pdfexport.pdf'
    };
    pivotTableObj.pdfExport(pdfExportProperties, false, null, true);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="pdf" type="button" value="PDF Export" name="PDF Export">
        <div id="PivotTable"></div>
        <br>
        <div id="PivotTable2"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Excel Exporting](#)

Globalization and localization in EJ2 JavaScript Pivotview control

Globalization is the combination of Internationalization and localization. You can adapt the component to various languages by parsing and formatting the date or number [Internationalization](#) & adding culture specific customization and translation to the text [Localization](#).

Internationalization

Internationalization library provides support for formatting and parsing the number, date, and time by using the official [Unicode CLDR](#) JSON data and also provides the `loadCldr` method to load the culture specific CLDR JSON data.

By default, all the Essential JS 2 component are specific to English culture ('en-US'). If you want to go with the different culture other than English, follow the below steps.

- Install the `CLDR-Data` package by using the below command (it installs the CLDR JSON data). For more information about CLDR-Data, refer to this [link](#).

```
npm install cldr-data --save
```

Once the package installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

- Now import the installed CLDR JSON data into the `app.ts` file. To import JSON data we need to install the JSON plugin loader. Here we have used the SystemJS JSON plugin loader.

```
npm install systemjs-plugin-json --save-dev
```

- Once installed, configure the `system.config.js` configuration settings as like below to map the `systemjs-plugin-json` loader.

```
`ts
System.config({
  paths: {
    'syncfusion:': './node_modules/@syncfusion/'
  },
  map: {
    app: 'app',
    //Syncfusion packages mapping
    '@syncfusion/ej2-base': 'syncfusion:ej2-base/dist/ej2-base.umd.min.js',
    '@syncfusion/ej2-data': 'syncfusion:ej2-data/dist/ej2-data.umd.min.js',
    '@syncfusion/ej2-inputs': 'syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js',
    '@syncfusion/ej2-buttons': 'syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js',
    '@syncfusion/ej2-splitbuttons': 'syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js',
    '@syncfusion/ej2-popups': 'syncfusion:ej2-popups/dist/ej2-popups.umd.min.js',
    '@syncfusion/ej2-navigations': 'syncfusion:ej2-navigations/dist/ej2-navigations.umd.min.js',
    '@syncfusion/ej2-grids': 'syncfusion:ej2-grids/dist/ej2-grids.umd.min.js',
    '@syncfusion/ej2-dropdowns': 'syncfusion:ej2-dropdowns/dist/ej2-dropdowns.umd.min.js',
    '@syncfusion/ej2-calendars': 'syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js',
    '@syncfusion/ej2-lists': 'syncfusion:ej2-lists/dist/ej2-lists.umd.min.js',
    '@syncfusion/ej2-excel-export': 'syncfusion:ej2-excel-export/dist/ej2-excel-export.umd.min.js',
    '@syncfusion/ej2-pdf-export': 'syncfusion:ej2-pdf-export/dist/ej2-pdf-export.umd.min.js',
    '@syncfusion/ej2-file-utils': 'syncfusion:ej2-file-utils/dist/ej2-file-utils.umd.min.js',
    '@syncfusion/ej2-compression': 'syncfusion:ej2-compression/dist/ej2-compression.umd.min.js',
```

```
"@syncfusion/ej2-pivotview": "syncfusion:ej2-pivotview/dist/ej2-pivotview.umd.min.js"
},
meta: {
  '*.json': { loader: 'plugin-json' }
},
packages: {
  'app': { main: 'app', defaultExtension: 'js' },
  'cldr-data': { main: 'index.js', defaultExtension: 'js' }
}
});
System.import('app');
`
```

- Now import the required culture from the installed location to `app.ts` file, like the below code snippets.

```
`ts
//import the loadCldr from ej2-base
import { loadCldr } from '@syncfusion/ej2-base';
loadCldr(
  require('cldr-data/supplemental/numberingSystems.json'),
  require('cldr-data/main/fr-CH/ca-gregorian.json'),
  require('cldr-data/main/fr-CH/numbers.json'));
require('cldr-data/main/fr-CH/timeZoneNames.json'));
`
```

The Internationalization library is used to globalize number, date, and time values in pivot table component using the `dataSourceSettings.formatSettings` option.

- Set the culture by using the `locale` property.

INDEX.TS

```
import { loadCldr, L10n, setCulture, setCurrencyCode } from
 '@syncfusion/ej2-base';
import * as currencies from './currencies.json';
import * as caGregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
import * as numberingSystems from './numberingSystems.json';
import { PivotView, IDataset, FieldList } from '@syncfusion/ej2-pivotview';
```

```

import { pivotData } from './datasource.ts';
loadCldr(currencies, cagregorian, numbers, timeZoneNames, numberingSystems);
setCulture('de');
setCurrencyCode('EUR');
L10n.load({
    'de-DE': {
        'pivotview': {
            'grandTotal': 'Gesamtsumme',
            'total': 'Insgesamt',
            'value': 'Wert',
            'noValue': 'Kein Wert',
            'row': 'Zeile',
            'column': 'Spalte',
            'collapse': 'Zusammenbruch',
            'expand': 'Erweitern'
        },
        "pivotfieldlist": {
            'fieldList': 'Feld Liste',
            'dropRowPrompt': 'Drop Reihe hier',
            'dropColPrompt': 'Drop column Hier',
            'dropValPrompt': 'Drop wert hier',
            'dropFilterPrompt': 'Drop Filter Hier',
            'addPrompt': 'Feld hinzufügen',
            'centerHeader': 'Ziehen Sie die Felder zwischen den Bereichen
unten:',
            'add': 'Hinzufügen',
            'drag': 'Ziehen',
            'filters': 'Filter',
            'rows': 'Zeilen',
            'columns': 'Spalten',
            'values': 'Werte',
            'error': 'Fehler',
            'dropAction': 'Berechnetes Feld nicht in jeder anderen Region
außer Wert Achse sein.',
            'search': 'Suche',
            'close': 'Schließen',
            'cancel': 'Abbrechen',
            'delete': 'Löschen',
            'alert': 'Warnung',
            'warning': 'Warnung',
            'ok': 'OK',
            'allFields': 'Alle Felder',
            'noMatches': 'Keine Treffer'
        }
    }
});
PivotView.Inject(FieldList);
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataSet[],
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],

```

```

        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C2', useGrouping: false,
            minimumSignificantDigits: 1, maximumSignificantDigits: 3,
            currency: 'EUR' }],
        filters: []
    },
    locale: 'de-DE',
    showFieldList: true,
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

* In the above sample, Amount field is formatted by [NumberFormatOptions](#). For date formats, the value strings are formatted by [DateFormatOptions](#).

* By default, locale value is en-US. If you want to change the en-US culture to a different culture, you have to change the locale accordingly.

* Also, you will find more details about support format string for number formats and data formats [here](#).

Decimal separators

<!-- markdownlint-disable MD009 -->

The decimal separators of pivot table values varies based on the culture applied to the component. The culture can be set by calling the method [setCulture](#) with appropriate culture string as its parameter.

The following example demonstrates the decimal separators in Deutsch culture.

INDEX.TS

```

import { loadCldr, L10n, setCulture, setCurrencyCode, Ajax } from
 '@syncfusion/ej2-base';
import * as currencies from './currencies.json';
import * as cagregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
import * as numberingSystems from './numberingSystems.json';
import { PivotView, IDataset, FieldList } from '@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
loadCldr(currencies, cagregorian, numbers, timeZoneNames, numberingSystems);
setCulture('de');
setCurrencyCode('EUR');
loadCultureFiles();
L10n.load({
    'de-DE': {
        'pivotview': {
            'grandTotal': 'Gesamtsumme',
            'total': 'Insgesamt',
            'value': 'Wert',
            'noValue': 'Kein Wert',
            'row': 'Zeile',
            'column': 'Spalte',
            'collapse': 'Zusammenbruch',
            'expand': 'Erweitern'
        }
    }
});

```



```

    },
    "pivotfieldlist": {
        'fieldList': 'Feld Liste',
        'dropRowPrompt': 'Drop Reihe hier',
        'dropColPrompt': 'Drop column Hier',
        'dropValPrompt': 'Drop wert hier',
        'dropFilterPrompt': 'Drop Filter Hier',
        'addPrompt': 'Feld hinzufügen',
        'centerHeader': 'Ziehen Sie die Felder zwischen den Bereichen
    unten:',
        'add': 'Hinzufügen',
        'drag': 'Ziehen',
        'filters': 'Filter',
        'rows': 'Zeilen',
        'columns': 'Spalten',
        'values': 'Werte',
        'error': 'Fehler',
        'dropAction': 'Berechnetes Feld nicht in jeder anderen Region
    außer Wert Achse sein.',
        'search': 'Suche',
        'close': 'Schließen',
        'cancel': 'Abbrechen',
        'delete': 'Löschen',
        'alert': 'Warnung',
        'warning': 'Warnung',
        'ok': 'OK',
        'allFields': 'Alle Felder',
        'noMatches': 'Keine Treffer'
    }
}
});
PivotView.Inject(FieldList);
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataset[],
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C2', currency: 'EUR' }],
        filters: []
    },
    locale: 'de-DE',
    showFieldList: true,
    height: 350
});
function loadCultureFiles() {
    var files = ['ca-gregorian.json', 'numbers.json', 'currencies.json',
'timeZoneNames.json', 'numberingSystems.json'];
    var loadCulture = function (prop: number) {
        var val, ajax;
        ajax = new Ajax('./' + files[prop], 'GET', false);
        ajax.onSuccess = function (value: any) {
            val = value;
        };
    };
}

```

```

        ajax.send();
        loadCldr(JSON.parse(val));
    };
    for (var prop = 0; prop < files.length; prop++) {
        loadCulture(prop);
    }
}
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>

```

```
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Localization

The [Localization](#) library allows you to localize default text content of the pivot table. The pivot table component has static text on some features (like drop area text, pivot field list title, etc...) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the

locale value and translation object.

The following list of properties and its values are used in the pivot table.

Locale keywords | Text

staticFieldList | Pivot Field List

fieldList | Field List

dropFilterPrompt | Drop filter here

dropColPrompt | Drop column here

dropRowPrompt | Drop row here

dropValPrompt | Drop value here

addPrompt | Add field here

adaptiveFieldHeader | Choose field

centerHeader | Drag fields between axes below:

add | add

drag | Drag

filter | Filter

filtered | Filtered

sort | Sort

remove | Remove

filters | Filters

rows | Rows

columns | Columns

values | Values

calculatedField | Calculated Field

createCalculatedField | Create Calculated Field

fieldName | Enter the field name

error | Error

invalidFormula | Invalid formula.

dropText | Example: ("Sum(OrderCount)" + "Sum(InStock)") * 250

dropTextMobile | Add fields and edit formula here.

dropAction | Calculated field cannot be place in any other region except value axis.

search | Search

close | Close

cancel | Cancel

delete | Delete

alert | Alert

warning | Warning

ok | OK

sum | Sum

average | Average

count | Count

min | Min

max | Max

allFields | All Fields

formula | Formula

fieldExist | A field already exists in this name. Please enter a different name.

confirmText | A calculation field already exists in this name. Do you want to replace it?

noMatches | No matches

format | Summaries values by

edit | Edit

clear | Clear

formulaField | Drag and drop fields to formula

dragField | Drag field to formula

Loading Translations

To load translation object in an application, use [load](#) function of the [L10n](#) class.

The following example demonstrates the pivot table in **Deutsch** culture.

INDEX.TS

```
import { L10n } from '@syncfusion/ej2-base';
import { PivotView, IDataset, FieldList } from '@syncfusion/ej2-pivotview';
```

```

import { pivotData } from './datasource.ts';
L10n.load({
  'de-DE': {
    'pivotview': {
      'grandTotal': 'Gesamtsumme',
      'total': 'Insgesamt',
      'value': 'Wert',
      'noValue': 'Kein Wert',
      'row': 'Zeile',
      'column': 'Spalte',
      'collapse': 'Zusammenbruch',
      'expand': 'Erweitern'
    },
    'pivotfieldlist': {
      'fieldList': 'Feld Liste',
      'dropRowPrompt': 'Drop Reihe hier',
      'dropColPrompt': 'Drop column Hier',
      'dropValPrompt': 'Drop wert hier',
      'dropFilterPrompt': 'Drop Filter Hier',
      'addPrompt': 'Feld hinzufügen',
      'centerHeader': 'Ziehen Sie die Felder zwischen den Bereichen
unten:',
      'add': 'Hinzufügen',
      'drag': 'Ziehen',
      'filters': 'Filter',
      'rows': 'Zeilen',
      'columns': 'Spalten',
      'values': 'Werte',
      'error': 'Fehler',
      'dropAction': 'Berechnetes Feld nicht in jeder anderen Region
außer Wert Achse sein.',
      'search': 'Suche',
      'close': 'Schließen',
      'cancel': 'Abbrechen',
      'delete': 'Löschen',
      'alert': 'Warnung',
      'warning': 'Warnung',
      'ok': 'OK',
      'allFields': 'Alle Felder',
      'noMatches': 'Keine Treffer'
    }
  }
});
PivotView.Inject(FieldList);
let pivotTableObj: PivotView = new PivotView({
  dataSourceSettings: {
    dataSource: pivotData as IDataSet[],
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  }
});

```

```

    },
    locale: 'de-DE',
    showFieldList: true,
    height: 350
  });
  pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Right-to-left (RTL)

RTL provides an option to switch the text direction and layout of the pivot table component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc...). To enable RTL pivot table, set the `enableRtl` property to **true**.

INDEX.TS

```

import { L10n } from '@syncfusion/ej2-base';
import { PivotView, IDataset, FieldList } from '@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
L10n.load({
    'ar-AE': {
        'pivotview': {
            'grandTotal': 'المجموع الكلي',
            'total': 'المجموع',
            'value': 'القيمة',
            'noValue': 'لا قيمة لها',
            'row': 'صف',
            'column': 'العمود',
            'collapse': 'الانهيـار',
            'expand': 'توسيع'
        },
        'pivotfieldlist': {
            'fieldList': 'قائمة الحقول',
            'dropRowPrompt': 'تراجع الخلاف هنا',
            'dropColPrompt': 'انخفاض العمود هنا',
            'dropValPrompt': 'انخفاض قيمة هنا',
            'dropFilterPrompt': 'انخفاض هنا عامل التصفية',
            'addPrompt': 'اضافة حقل هنا',
            'adaptiveFieldHeader': 'اختر الحقل',
            'centerHeader': ': اسحب المجالات بين المناطق الموضحة ادناه',
            'add': 'اضافة',
            'drag': 'اسحب',
            'filters': 'عوامل التصفية',
            'rows': 'الصفوف',
            'columns': 'الاعمدة',
            'values': 'قيم',
            'search': 'البحث',
            'close': 'قريب',
            'cancel': 'الغاء',
            'delete': 'احذف',
            'alert': 'حالة تاهب قصوى',
            'warning': 'تحذير',
            'ok': 'موافق',
            'allFields': 'جميع الحقول',
            'noMatches': 'لا مباريات'
        }
    }
});

```

```

});
PivotView.Inject(FieldList);
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataset[],
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    enableRtl: true,
    locale: 'ar-AE',
    showFieldList: true,
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

```



```

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Internationalization](#)
- [Localization](#)

Accessibility in Javascript Pivotview component

The pivot table component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the pivot table component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

```

| Mobile Device Support |  |
| Keyboard Navigation Support |  |
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The component does not meet the requirement.</div>

```

WAI-ARIA attributes

[WAI-ARIA](#) (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components. The following ARIA attributes are used in the pivot table component:

Attributes	Purpose
---	---
role=grid	Attribute added to identify the grid component element within the pivot table element.
role=region	Attribute added to identify the chart component element within the pivot table element.
role=button	This attribute is added to the pager navigation buttons as well as the buttons in the dialog popup such as field list, calculated field, member editor, conditional formatting of pivot table component to indicate that it is a clickable element.
role=table	This attribute is added to each conditional formatting style container element to denote it as a table.
role=tableItems	This attribute is added to the container element that appears inside the number formatting popup to indicate it as a table.

| **aria-disabled** | The buttons within the dialog popups, such as field list, calculated field and member editor, will be disabled based on their usability. To indicate its disabled state, we will add this attribute with the values **true**. By default, the attribute value is set to **false**. |

| **aria-label** | This attribute is added to label elements that are placed inside the pager, member editor popup, and calculated field popup to identify them as label elements. |

| **aria-selected** | This attribute is added to the selected treeview item in the calculated field popup with the value as **true** to denote that it is a selected element. |

| **aria-colspan** | This attribute is added to the **th** elements in the **e-table**, which represent the column span value. |

| **aria-rowspan** | This attribute is added to the **th** elements in the **e-table**, which represent the row span value. |

| **data-type** | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It represents the aggregate type for the specified field. |

| **data-caption** | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It represents the caption for the specified field. |

| **data-basefield** | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It denotes the base field for the specified field, which is used to display the values for aggregation types such as **DifferenceFrom**, **PercentageOfDifferenceFrom**, and **PercentageOfParentTotal**. |

| **data-baseitem** | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It denotes the base item for the specified field, which is used to display the values for aggregation types such as **DifferenceFrom**, **PercentageOfDifferenceFrom**, and **PercentageOfParentTotal**. |

| **data-field** | This attribute is added to the treeview item in the calculated field popup. It denotes the name of the specified field. |

| **data-membertype** | This attribute is added to the treeview item in the calculated field popup. It denotes the member type of the selected OLAP calculated field. |

| **data-hierarchy** | This attribute is added to the treeview item in the calculated field popup. It denotes the parent hierarchy unique name of the selected OLAP calculated field. |

| **data-formula** | This attribute is added to the treeview item in the calculated field popup. It denotes the formula used for the specified calculated field. |

| **data-formatString** | This attribute is added to the treeview item in the calculated field popup. It denotes the format string used for the specified calculated field. |

| **data-customformatstring** | This attribute is added to the treeview item in the calculated field popup. It denotes the custom format string used for the specified calculated field. |

Keyboard interaction

The pivot table component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Message component.

| **Press** | **To do this** |

| --- | --- |

| **Tab / Shift + Tab** | **To focus the close icon in the message.** |

| **Enter / Space** | **Closes the focused close icon's message.** |

Pivot Table

| **Press** | **To do this** |

| --- | --- |

| **Tab** | **Moves the cell focus right side. If no cells are focused, it moves to the next active element in the browser page.** |

| **Shift + Tab** | **Moves the cell focus left side. If no cells are focused, it moves to the previous active element in the browser page.** |

| **DownArrow** | **Moves the cell focus downwards. If the selection is enabled in the pivot table, then it will move downwards to select next row or column or individual cell.** |

| **UpArrow** | **Moves the cell focus upwards. If the selection is enabled in the pivot table, then it will move upwards to select previous row or column or individual cell.** |

| **LeftArrow** | **Moves the cell focus left side. If the selection is enabled in the pivot table, then it will move left side to select previous row or column or individual cell.** |

| **RightArrow** | **Moves the cell focus right side. If the selection is enabled in the pivot table, then it will move right side to select next row or column or individual cell.** |

| **Shift + DownArrow** | **Extends the cell selection downwards.** |

| **Shift + UpArrow** | **Extends the cell selection selection upwards.** |

| **Shift + LeftArrow** | **Extends the cell selection to the left side.** |

| **Shift + RightArrow** | **Extends the cell selection to the right side.** |

| **Ctrl + A** | **Selects all cells.** |

| **Esc** | **Deselects all cells. If the current active element is a context menu, then the context menu popup will be closed.** |

| **Home** | **Goes to the first cell in the current row.** |

| **End** | **Goes to the last cell in the current row.** |

| **Ctrl + Home** | **Goes to the first cell in the table.** |

| **Ctrl + End** | **Goes to the last cell in the table.** |

| **Enter** | **If the current cell is an expand/collapse cell, it performs expand/collapse operation (drill operation). If the current row/column header is in value sort state, it performs value sorting. If the current cell is in selection state, it moves to the next row, column or individual cell. If drill-through or editing is enabled in the pivot table, the drill-through dialog will be**

opened based on the selected value cell. If the current active element is a context menu popup, menu selection will be performed. |

| Shift + Enter | If value sorting is enabled in the pivot table and the current cell is a header with respect to its value axis, it performs value sorting to either ascending or descending order. If the current cell is in selection state, it moves to the previous row, column or individual cell. |

| Ctrl + Enter | If hyperlink is enabled in the current cell, it performs hyperlink selection. |

| Shift + F10 or Menu | If context menu is enabled in the pivot table, the context menu popup will be opened in the current cell. |

Field List

| Press | To do this |

| --- | --- |

| Shift + Ctrl + F | If the popup field list is enabled in either the pivot table or the pivot chart, the field list dialog will be opened. |

| Tab | Moves to the next active element in the field list. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the field list. If no active elements present, it moves to the previous active element in the browser page. |

| Shift + F | If the current active element is a field's button and if it has a filter icon, the filter dialog will open to perform filtering. |

| Shift + S | If the current active element is a field's button and if it has a sort icon, the sorting will be performed to the selected field. |

| Shift + E | If the current active element is a calculated field's button and if it has an edit icon, the calculated field dialog will be opened to perform editing the selected calculated field. |

| Enter | Performs the selection operation of the current active element. If the current active element is a field's button and it has a dropdown icon, the aggregation menu will open to perform calculations using aggregation options to the selected value field. |

| Delete | If the current active element is a field's button, the selected field will be removed from the current report. |

| DownArrow | If the current active element is a tree node, it moves to the next node. |

| UpArrow | If the current active element is a tree node, it moves to the previous node. |

| LeftArrow | If the current active element is a tree node, it collapses the current node. |

| RightArrow | If the current active element is a tree node, it expands the current node. |

| Home | If the current active element is a tree node, it goes to the first node. |

| End | If the current active element is a tree node, it goes to the last node. |

| Space | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked. |

| Esc or Escape | Closes the popup field list dialog. |

Grouping Bar

| Press | To do this |

| --- | --- |

| Tab | Moves to the next active element (field's button) in the grouping bar. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element (field's button) in the grouping bar. If no active elements present, it moves to the previous active element in the browser page. |

| Shift + F | If the current active element is a field's button and if it has a filter icon, the filter dialog will be opened to perform filtering. |

| Shift + S | If the current active element is a field's button and if it has a sort icon, the sorting will be performed to the selected field. |

| Shift + E | If the current active element is a calculated field's button and if it has an edit icon, the calculated field dialog will be opened to perform editing the selected calculated field. |

| Enter | Performs the selection operation of the current active element. If the current active element is a field's button and if it has a dropdown icon, the aggregation menu will be opened to perform calculations using aggregation options to the selected value field. |

| Delete | If the current active element is a field's button, the selected field will be removed from the current report. |

| DownArrow | If the current active element is a dropdown list, the next item will be selected. |

| UpArrow | If the current active element is a dropdown list, the previous item will be selected. |

| Home | If the current active element is a dropdown list, the first item will be selected. |

| End | If the current active element is a dropdown list, the last item will be selected. |

| Alt + Down | If the current active element is a dropdown list, the popup will be opened. |

| Alt + Down | If the current active element is a dropdown list, the popup will be closed. |

| Esc or Escape | Closes the dropdown list.

Filter Dialog

| Press | To do this |

| --- | --- |

| Shift + F | If the current active element is a field's button and if it has a filter icon in either the field list or grouping bar UI, the filter dialog will be opened to perform filtering. |

| Tab | Moves to the next active element in the filter dialog. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the filter dialog. If no active elements present, it moves to the previous active element in the browser page. |

| DownArrow | If the current active element is a tree node, it moves to the next node. |

| UpArrow | If the current active element is a tree node, it moves to the previous node. |

| LeftArrow | If the current active element is a tree node, it collapses the current node. If the current active element is a tab, it moves focus to the previous tab element. |

| RightArrow | If the current active element is a tree node, it expands the current node. If the current active element is a tab, it moves focus to the next tab element. |

| Home | If the current active element is a tree node, it goes to the first node. |

| End | If the current active element is a tree node, it goes to the last node. |

| Space | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked. |

| Alt + Down | If the current active element is a DropDownList or DatePicker or DateTimePicker, the popup will be opened. |

| Alt + Up | If the current active element is a DropDownList or DatePicker or DateTimePicker, the popup will be closed. |

| Enter | Performs the selection operation of the current active element. If the current active element is a tab, the current tab element will be selected. If the current active element is a tree node, the current node will be either checked or unchecked. If the current active element is DropDownList, the focus item will be selected, and the popup list will close when it is open. Otherwise, toggles the popup list. |

| Esc or Escape | Closes the filter dialog. |

Calculated Field Dialog

| Press | To do this |

| --- | --- |

| Shift + E | If the current active element is a field's button and if it has an edit icon in either the field list or grouping bar UI, the calculated field dialog will be opened to perform editing the selected calculated field. |

| Tab | Moves to the next active element in the calculated field dialog. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the calculated field dialog. If no active elements present, it moves to the previous active element in the browser page. |

| DownArrow | If the current active element is a tree node, it moves to the next node. |

| UpArrow | If the current active element is a tree node, it moves to the previous node. |

| LeftArrow | If the current active element is a tree node, it collapses the current node. |

| RightArrow | If the current active element is a tree node, it expands the current node. If the current active element is a tree node and has a menu icon, the aggregation menu will be opened to select appropriate aggregation type to the selected field. |

| Home | If the current active element is a tree node, it goes to the first node. |

| End | If the current active element is a tree node, it goes to the last node. |

| Enter | Performs the selection operation of the current active element. If the current active element is a tree node, it copies the selected field name/formula to the formula text area to perform calculations. |

| Esc or Escape | Closes the calculated field dialog. |

Formatting Dialog

| Press | To do this |

| --- | --- |

| Tab | Moves to the next active element in the formatting dialog. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the formatting dialog. If no active elements present, it moves to the previous active element in the browser page. |

| DownArrow | If the current active element is a DropDownList, the next item will be selected. |

| UpArrow | If the current active element is a DropDownList, the previous item will be selected. |

| Home | If the current active element is a DropDownList, the first item will be selected. |

| End | If the current active element is a DropDownList, the last item will be selected. |

| Alt + Down | If the current active element is a DropDownList or ColorPicker, the popup will be opened. |

| Alt + Down | If the current active element is a DropDownList or ColorPicker, the popup will be closed. |

| Enter | Performs the selection operation of the current active element. |

| Esc or Escape | Closes the formatting dialog. |

Toolbar

| Press | To do this |

| --- | --- |

| Tab | Moves to the next active option in the toolbar. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active option in the toolbar. If no active elements present, it moves to the previous active element in the browser page. |

| Enter | Performs the selection operation of the current active element. |

Drill-Through Dialog

| Press | To do this |

| --- | --- |

| Tab | Moves to the next active element in the drill-through dialog. If the current active element is a Grid cell, it moves the cell focus to right side. If no active elements present, then it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the drill-through dialog. If the current active element is a Grid cell, it moves the cell focus to left side, If no active elements present, then it moves to the previous active element in the browser page. |

| DownArrow | Moves the row/cell focus downwards. |

| UpArrow | Moves the row/cell focus upwards. |

| LeftArrow | Moves the cell focus left side. |

| RightArrow | Moves the cell focus right side. |

| Home | Goes to the first cell in the current row. |

| End | Goes to the last cell in the current row. |

| Ctrl + Home | Goes to the first cell in the table. |

| Ctrl + End | Goes to the last cell in the table. |

| Enter | Performs the selection operation of the current active element. |

| Esc or Escape | If the cell is in selected state, the it deselects all rows/cells. If the row/cell is in edit state, it cancels the current entries in the row/cell. If the current active element is not a row/cell, it closes the drill-through dialog. |

| F2 | Initiate editing a row/cell in the data grid. |

| Insert | Adds a new row/cell in the data grid. |

| Delete | Removes the selected row in the data grid. |

Some commonly used applicable key combinations and their relative functionalities in all dialogs are listed below.

| Press | To do this |

| --- | --- |

| Tab | Moves to the next active element in the dialog. If either no active elements present in the dialog or an overlay is not present in the dialog, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the dialog. If either no active elements present in the dialog or an overlay is not present in the dialog, it moves to the previous active element in the browser page. |

| Space | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked. |

| Enter | When the Dialog button or any input (except text area) is in focus state, when pressing the Enter key, the click event associated with the primary button or button will be triggered. The Enter key will not be worked, when the dialog content contains any text area with initial focus. |

| Esc or Escape | Closes the dialog. |

Ensuring accessibility

The pivot table component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the pivot table component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the pivot table component with accessibility tools.

INDEX.JS

```
var pivotTableObj = new ej.pivotview.PivotView({
  dataSourceSettings: {
    dataSource: pivotData,
    expandAll: true,
    enableSorting: true,
    allowLabelFilter: true,
    allowValueFilter: true,
    sortSettings: [{ name: 'company', order: 'Descending' }],
    formatSettings: [{ name: 'balance', format: 'C' }, { name: 'date',
format: 'dd/MM/yyyy-hh:mm', type: 'date' }],
    drilledMembers: [{ name: 'product', items: ['Bike', 'Car'] }, {
name: 'gender', items: ['male'] }],
    filterSettings: [
      { name: 'date', type: 'Date', condition: 'Between', value1: new
Date('02/16/2000'), value2: new Date('02/16/2002') },
      { name: 'age', type: 'Number', condition: 'Between', value1:
'25', value2: '35' },
      { name: 'eyeColor', type: 'Exclude', items: ['blue'] }
    ],
    rows: [{ name: 'state' }, { name: 'eyeColor' }],
    columns: [{ name: 'gender', caption: 'Population' }, { name:
'isActive' }],
    values: [{ name: 'balance' }, { name: 'quantity' }],
    filters: [],
    conditionalFormatSettings: [
      {
        measure: 'balance',
        value1: 100000,
        conditions: 'LessThan',
        style: {
          backgroundColor: '#80cbc4',
          color: 'black',
          fontFamily: 'Tahoma',
          fontSize: '12px'
        }
      },
      {
        value1: 10,
        value2: 20,
```

```

        measure: 'quantity',
        conditions: 'Between',
        style: {
            backgroundColor: '#f48fb1',
            color: 'black',
            fontFamily: 'Tahoma',
            fontSize: '12px'
        }
    }
    ],
    showGroupingBar: true,
    groupingBarSettings: { showFieldsPanel: true },
    displayOption: { view: 'Both' },
    chartSettings: {
        value: 'Amount', enableExport: true, chartSeries: { type: 'Column',
animation: { enable: false } }, enableMultipleAxis: false,
    },
    toolbar: ['New', 'Save', 'SaveAs', 'Rename', 'Remove', 'Load',
        'Grid', 'Chart', 'MDX', 'Export', 'SubTotal', 'GrandTotal',
'Formatting', 'FieldList'],
    editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Normal' },
    allowExcelExport: true,
    allowConditionalFormatting: true,
    allowNumberFormatting: true,
    allowPdfExport: true,
    allowGrouping: true,
    showToolbar: true,
    allowCalculatedField: true,
    showFieldList: true,
    allowDeferLayoutUpdate: true,
    saveReport: function (args) {
        var reports = [];
        var isSaved = false;
        if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
            reports = JSON.parse(localStorage.pivotviewReports);
        }
        if (args.report && args.reportName && args.reportName != '') {
            reports.map(function (item) {
                if (args.reportName === item.reportName) {
                    item.report = args.report;
                    isSaved = true;
                }
            });
            if (!isSaved) {
                reports.push(args);
            }
            localStorage.pivotviewReports = JSON.stringify(reports);
        }
    },
    fetchReport: function (args) {
        var reportCollection = [];
        var reeportList = [];
        if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {

```

```

        reportCollection = JSON.parse(localStorage.pivotviewReports);
    }
    reportCollection.map(function (item) {
        reeportList.push(item.reportName);
    });
    args.reportName = reeportList;
},
loadReport: function (args) {
    var reportCollection = [];
    if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "" ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
    }
    reportCollection.map(function (item) {
        if (args.reportName === item.reportName) {
            args.report = item.report;
        }
    });
    if (args.report) {
        pivotTableObj.dataSourceSettings =
JSON.parse(args.report).dataSourceSettings;
    }
},
removeReport: function (args) {
    var reportCollection = [];
    if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "" ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
    }
    for (var i = 0; i < reportCollection.length; i++) {
        if (reportCollection[i].reportName === args.reportName) {
            reportCollection.splice(i, 1);
        }
    }
    if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "" ) {
        localStorage.pivotviewReports =
JSON.stringify(reportCollection);
    }
},
renameReport: function (args) {
    var reportCollection = [];
    if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "" ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
    }
    if (args.isReportExists) {
        for (var i = 0; i < reportCollection.length; i++) {
            if (reportCollection[i].reportName === args.rename) {
                reportCollection.splice(i, 1);
            }
        }
    }
    reportCollection.map(function (item) {
        if (args.reportName === item.reportName) {
            item.reportName = args.rename;
        }
    });
}

```

```

    });
    if (localStorage.pivotviewReports && localStorage.pivotviewReports
    !== "") {
        localStorage.pivotviewReports =
        JSON.stringify(reportCollection);
    }
},
newReport: function () {
    pivotTableObj.setProperties({
        dataSourceSettings: {
            columns: [],
            rows: [],
            values: [],
            filters: []
        }
    }, false);
},
toolbarRender: function (args) {
    args.customToolbar.splice(6, 0, {
        type: 'Separator'
    });
    args.customToolbar.splice(9, 0, {
        type: 'Separator'
    });
},
gridSettings: {
    columnWidth: 140,
    contextMenuItems: [
        'Aggregate', 'CalculatedField', 'Drillthrough', 'Excel Export',
        'Pdf Export', 'Csv Export', 'Expand',
        'Collapse', 'Sort Ascending', 'Sort Descending'
    ]
},
width: '100%',
height: 450
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Pivot Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material3.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material3.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material3.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material3.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material3.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material3.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material3.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material3.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material3.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material3.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material3.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if(ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

Ej1 api migration in EJ2 JavaScript Pivotview control

This article describes the API migration process of pivot table component from Essential JS 1 to Essential JS 2.

Data Binding

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Data source | **property:** dataSource

\$("#PivotGrid").ejPivotGrid({
dataSource:
data: []
}); | **property:** dataSourceSettings

var pivotGridObj = new

```
ej.pivotview.PivotView({<br/>dataSourceSettings: {<br/>dataSource:
[<br/>}}];<br/>pivotGridObj.appendTo('#PivotGrid');|
```

```
| Rows | property: rows<br/><br/>$("#PivotGrid").ejPivotGrid({<br/>dataSource: {<br/>rows:
[{fieldName: "Country",fieldCaption: "Country"}] }<br/>}); | property: rows<br/><br/>var pivotGridObj =
new ej.pivotview.PivotView({<br/> dataSourceSettings: {<br/>rows: [{ name: 'company', caption:
'Industry' }]}<br/>});<br/>pivotGridObj.appendTo('#PivotGrid');|
```

```
| Columns | property: columns<br/><br/>$("#PivotGrid").ejPivotGrid({<br/> dataSource: {<br/>columns:
[ { fieldName: "Country",fieldCaption: "Country" } ] }<br/>}); | property: columns<br/><br/>var
pivotGridObj = new ej.pivotview.PivotView({<br/> dataSourceSettings: {<br/> columns: [{ name:
'company', caption: 'Industry' }]}<br/>});<br/>pivotGridObj.appendTo('#PivotGrid');|
```

```
| Values | property: values<br/><br/>$("#PivotGrid").ejPivotGrid({<br/> dataSource: {<br/> values: [ {
fieldName: "balance", fieldCaption: "Balance($)" } ] }<br/> }); | property: values<br/><br/>var
pivotGridObj = new ej.pivotview.PivotView({<br/>dataSourceSettings: {<br/>values: [{ name: 'balance',
caption: 'Balance($)' }]}<br/>});<br/>pivotGridObj.appendTo('#PivotGrid');|
```

```
| Filters | property: filters<br/><br/>$("#PivotGrid").ejPivotGrid({<br/> dataSource: {<br/>filters: [ {
fieldName: "Country", fieldCaption: "Country" } ] }<br/> }); | property: filters<br/><br/>var pivotGridObj
= new ej.pivotview.PivotView({<br/> dataSourceSettings: {<br/>filters: [{ name: 'company', caption:
'Industry' }]}<br/>});<br/>pivotGridObj.appendTo('#PivotGrid');|
```

```
| Value axis position | Not Applicable | property: valueAxis<br/><br/>var pivotGridObj = new
ej.pivotview.PivotView({<br/>dataSourceSettings: { valueAxis:
'row'}<br/>});<br/>pivotGridObj.appendTo('#PivotGrid');|
```

Aggregation

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

```
| Summary Type | property: summaryType<br/><br/>$("#PivotGrid").ejPivotGrid({<br/> dataSource:
{<br/>values: [ {<br/> fieldName: "balance",<br/>fieldCaption: "Balance($)",<br/>summaryType:
ej.PivotAnalysis.SummaryType.Count<br/> } ] }<br/> }); | property: type<br/><br/>var pivotGridObj =
new ej.pivotview.PivotView({<br/> dataSourceSettings: {<br/> values: [{ name: 'balance', caption:
'Balance($)', type: 'Count' }]}<br/>});<br/>pivotGridObj.appendTo('#PivotGrid');|
```

Number Format

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

```
| Format settings | property: format<br/><br/>$("#PivotGrid").ejPivotGrid({<br/> dataSource: {<br/>
values: [ { fieldName: "balance", fieldCaption: "Balance($)", format: "currency" } ] }<br/> }); | property:
formatSettings<br/><br/>var pivotGridObj = new ej.pivotview.PivotView({<br/> dataSourceSettings:
{<br/>formatSettings: [{ name: 'balance', format: 'C' },<br/> { name: 'date', format: 'dd/MM/yyyy-
hh:mm', type: 'date' }]}<br/>});<br/>pivotGridObj.appendTo('#PivotGrid');|
```

Summary Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show/hide grand totals | **property:** enableGrandTotal

\$("#PivotGrid").ejPivotGrid({
enableGrandTotal: false
 }); | **property:** showGrandTotals

var pivotGridObj = new ej.pivotview.PivotView({
dataSourceSettings: {
showGrandTotals: false
}
});
pivotGridObj.appendTo('#PivotGrid');|

| Show/hide row grand totals | **property:** enableRowGrandTotal

\$("#PivotGrid").ejPivotGrid({
enableRowGrandTotal: false
}); | **property:** showRowGrandTotals

var pivotGridObj = new ej.pivotview.PivotView({
dataSourceSettings: {
showRowGrandTotals: false
}
});
pivotGridObj.appendTo('#PivotGrid');|

| Show/hide column grand totals | **property:** enableColumnGrandTotal

\$("#PivotGrid").ejPivotGrid({
enableColumnGrandTotal: false
 }); | **property:** showColumnGrandTotals

var pivotGridObj = new ej.pivotview.PivotView({
dataSourceSettings: {
showColumnGrandTotals: false
}
});
pivotGridObj.appendTo('#PivotGrid');|

| Show/hide sub-totals | Not Applicable | **property:** showSubTotals

var pivotGridObj = new ej.pivotview.PivotView({
dataSourceSettings: {
showSubTotals: false
}
});
pivotGridObj.appendTo('#PivotGrid');|

| Show/hide row sub-totals | Not Applicable | **property:** showRowSubTotals

var pivotGridObj = new ej.pivotview.PivotView({
dataSourceSettings: {
showRowSubTotals: false
}
});
pivotGridObj.appendTo('#PivotGrid');|

| Show/hide column sub-totals | Not Applicable | **property:** showColumnSubTotals

var pivotGridObj = new ej.pivotview.PivotView({
dataSourceSettings: {
showColumnSubTotals: false
}
});
pivotGridObj.appendTo('#PivotGrid');|

| Show/hide sub-totals for specific field | **property:** showSubTotal

new ej.PivotGrid(\$("#PivotGrid"){
dataSource: {
rows: [{ fieldName: "Country", showSubTotal : false}] }
}); | **property:** showSubTotals

var pivotGridObj = new ej.pivotview.PivotView({
dataSourceSettings: {
rows: [{ name: 'company', showSubTotals: false }] }
});
pivotGridObj.appendTo('#PivotGrid');|

Drill operation

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Expand All | **property:** enableCollapseByDefault

\$("#PivotGrid").ejPivotGrid({
enableCollapseByDefault:true
 }); | **property:** expandAll

var pivotGridObj = new ej.pivotview.PivotView({
dataSourceSettings: {
expandAll: false
}
});
pivotGridObj.appendTo('#PivotGrid');|

| Drill Up/Down | **property:** collapsedMembers

\$("#PivotGrid").ejPivotGrid({
collapsedMembers: { Country: ["Canada", "France"] }
}); | **property:** drilledMembers

var pivotGridObj = new ej.pivotview.PivotView({
dataSourceSettings: {
drilledMembers: [{ name: 'Country', items: ['France'] }] }
}
});
pivotGridObj.appendTo('#PivotGrid');|

Field List

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show/hide field list pop-up button on pivot table | Not Applicable | **property:**

```
showFieldList<br><br>var pivotGridObj = new ej.pivotview.PivotView({<br> showFieldList:
true<br>});<br>pivotGridObj.appendTo('#PivotGrid');|
```

| Defer update | **property:** enableDeferUpdate

\$("#PivotGrid").ejPivotGrid({

enableDeferUpdate: true
}); | Not Applicable |

| Control initialization | **component:**

```
PivotSchemaDesigner<br><br>$("#PivotSchemaDesigner").ejPivotSchemaDesigner({});<br>| compone
nt: PivotFieldList<br><br>var fieldlistObj = new
ej.pivotview.PivotFieldList({});<br>fieldlistObj.appendTo('#FieldList');|
```

| Render mode | Not Applicable | **property:** renderMode

var fieldlistObj = new
ej.pivotview.PivotFieldList({
renderMode: 'Fixed'});
fieldlistObj.appendTo('#FieldList');|

| Show/hide calculated field button | Not Applicable | **property:** allowCalculatedField

var
fieldlistObj = new ej.pivotview.PivotFieldList({
 allowCalculatedField:
true
});
fieldlistObj.appendTo('#FieldList');|

| Show/hide value group button | Not Applicable | **property:** showValuesButton

var fieldlistObj
= new ej.pivotview.PivotFieldList({
showValuesButton:
true
});
fieldlistObj.appendTo('#FieldList');|

Grouping Bar

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show/hide Grouping bar | **property:** enableGroupingBar

\$("#PivotGrid").ejPivotGrid({

enableGroupingBar: true
}); | **property:** showGroupingBar

var pivotGridObj = new
ej.pivotview.PivotView({
 showGroupingBar:
true
});
pivotGridObj.appendTo('#PivotGrid');|

| Grouping Bar Settings | Not Applicable | **property:** groupingBarSettings

var pivotGridObj = new
ej.pivotview.PivotView({
 groupingBarSettings: {

showFilterIcon: false,
showSortIcon:
true,
showRemoveIcon: false}
});
pivotGridObj.appendTo('#PivotGrid');|

| Show/hide value group button | Not Applicable | **property:** showValuesButton

var
pivotGridObj = new ej.pivotview.PivotView({
showValuesButton:
true
});
pivotGridObj.appendTo('#PivotGrid');|

Filtering

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Filter settings | **property:** filterItems

\$("#PivotGrid").ejPivotGrid({
 dataSource: {

rows: [{fieldName: "country",

filterItems : {
 filterType:
ej.PivotAnalysis.FilterType.Exclude,
 values: ["Canada", "France"] }
 }] }
}); | **property:**
filterSettings

var pivotGridObj = new ej.pivotview.PivotView({
 dataSourceSettings:
{
 filterSettings: [{
 name: 'eyeColor',
 type: 'Exclude', items: ['blue']
}]
});
pivotGridObj.appendTo('#PivotGrid');|

Maximum node limit in member editor

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Max node limit in member editor | **property:**

```
maxNodeLimitInMemberEditor<br/><br/>$("#PivotGrid").ejPivotGrid({<br/>
maxNodeLimitInMemberEditor: 100<br/>}); | property: maxNodeLimitInMemberEditor<br/><br/>var
pivotGridObj = new ej.pivotview.PivotView({<br/> maxNodeLimitInMemberEditor:
100<br/>});<br/>pivotGridObj.appendTo('#PivotGrid');|
```

No Data Items

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show/hide "no data" items | Not Applicable | **property:** showNoDataItems

var pivotGridObj =
new ej.pivotview.PivotView({
 dataSourceSettings: {
 rows: [{
 name: 'eyeColor',
showNoDataItems: true }]
});
pivotGridObj.appendTo('#PivotGrid');|

Excel-like filtering

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Label filtering | **property:** enableAdvancedFilter

\$("#PivotGrid").ejPivotGrid({

enableAdvancedFilter: true,
 dataSource: {
 rows: [{ fieldName: "country",

advancedFilter : [{
 labelFilterOperator: ej.olap.LabelFilterOptions.EndsWith,
 values: ["es"]
}]
 }] }
}); | **property:** allowLabelFilter

var pivotGridObj = new
ej.pivotview.PivotView({
dataSourceSettings: {
 allowLabelFilter: true,
 filterSettings: [{

 name: 'product',
 type: 'Label',
 condition: 'Between',
 value1: 'e',
value2: 'v'
}]
});
pivotGridObj.appendTo('#PivotGrid');|

| Value filtering | **property:** enableAdvancedFilter

\$("#PivotGrid").ejPivotGrid({

enableAdvancedFilter: true,
 dataSource: {
 rows: [{ fieldName: "country",
 advancedFilter
: [{
 measure: "balance",
valueFilterOperator: ej.olap.ValueFilterOptions.GreaterThan,

values: ["200"] }]
 }]
}); | **property:** allowValueFilter

var pivotGridObj = new
ej.pivotview.PivotView({
dataSourceSettings: {
allowValueFilter: true,
filterSettings: [{

 name: 'product',
 measure: 'quantity',
 type: 'Value',
condition: 'Between',

value1: '3250',
value2: '5000' }]
});
pivotGridObj.appendTo('#PivotGrid');|

Sorting

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Enable/disable sorting | Not Applicable | **property:** enableSorting

var pivotGridObj = new
ej.pivotview.PivotView({
 dataSourceSettings: {
 enableSorting:
false
});
pivotGridObj.appendTo('#PivotGrid');|

| Sort settings | **property:** sortOrder

\$("#PivotGrid").ejPivotGrid({
dataSource: {

rows: [{
 fieldName: "Country",
 sortOrder : ej.PivotAnalysis.SortOrder.Descending }]
 }
});
 | **property:** sortSettings

var pivotGridObj = new ej.pivotview.PivotView({

dataSourceSettings: {
 sortSettings: [{
 name: 'company',
order: 'Descending' }]

});
pivotGridObj.appendTo('#PivotGrid');|

Value Sorting

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Enable/disable value sorting | Not Applicable | **property:** enableValueSorting

var pivotGridObj = new ej.pivotview.PivotView({
 enableValueSorting: true
});
pivotGridObj.appendTo('#PivotGrid');|

| Value sort settings | **property:** valueSortSettings

\$("#PivotGrid").ejPivotGrid({
valueSortSettings: {
 headerText: "Bike##Quantity",
 headerDelimiters: "##",
sortOrder: ej.PivotAnalysis.SortOrder.Descending }
});| **property:** valueSortSettings

var pivotGridObj = new ej.pivotview.PivotView({
 dataSourceSettings: {
valueSortSettings: {
headerText: 'FY 2015##Sold Amount',
headerDelimiter: '##',
 sortOrder: 'Descending' }
});
pivotGridObj.appendTo('#PivotGrid');|

Calculated Field

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show/hide calculated field | Not Applicable | **property:** allowCalculatedField

var pivotGridObj = new ej.pivotview.PivotView({
 allowCalculatedField: true
});
pivotGridObj.appendTo('#PivotGrid');|

| Calculated field settings | **property:** values

\$("#PivotGrid").ejPivotGrid({
 dataSource: {
values: [{
 fieldName: "Amount", fieldCaption: "Amount"
}, {
 fieldName: "Price",
fieldCaption: "Price",
 isCalculatedField: true,
 formula: "Amount*15" }]
});| **property:** calculatedFieldSettings

var pivotGridObj = new ej.pivotview.PivotView({
dataSourceSettings: {
values: [{
 name: 'Total', type: 'CalculatedField' }],
 calculatedFieldSettings: [{
 name: 'Total',
 formula: "Sum(Amount)"+"Sum(Sold)" }]}
});
pivotGridObj.appendTo('#PivotGrid');|

Paging

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Paging | **property:** enablePaging

\$("#PivotGrid").ejPivotGrid({
 enablePaging : true
});| Not Applicable |

| Virtual scrolling | **property:** enableVirtualScrolling

\$("#PivotGrid").ejPivotGrid({
 enableVirtualScrolling : true
 });| **property:** enableVirtualization

var pivotGridObj = new ej.pivotview.PivotView({
 enableVirtualization: true});
pivotGridObj.appendTo('#PivotGrid');|

Conditional Formatting

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show/hide conditional formatting | **property:** enableConditionalFormatting

\$("#PivotGrid").ejPivotGrid({
 enableConditionalFormatting: true
 });| **property:** allowConditionalFormatting

var pivotGridObj = new ej.pivotview.PivotView({
 allowConditionalFormatting: true
});
pivotGridObj.appendTo('#PivotGrid');|

| Conditional formatting settings | **property:**
conditionalFormatSettings

\$("#PivotGrid").ejPivotGrid({
conditionalFormatSettings:
[
name: "Format2",
 style: {
"color": "#000000",
 "backgroundColor":
"#0000FF",
 "bordercolor": "#000000",
"borderstyle": "Dashed",
 "borderwidth":
"5",
 "fontSize": "12",
 "fontstyle": "Algerian" },
condition:
ej.PivotGrid.ConditionalOptions.LessThan,
 value: "200",
 measures: "Amount,Quantity"
}]
}); | **property:** conditionalFormatSettings

var pivotGridObj = new
ej.pivotview.PivotView({
 dataSourceSettings: {
conditionalFormatSettings: [{
 measure:
'In Stock',
value1: 5000,
 conditions: 'LessThan',
 style: {
 backgroundColor:
'#80cbc4',
 color: 'black',
 fontFamily: 'Tahoma',
fontSize: '12px' }

}]
});
pivotGridObj.appendTo('#PivotGrid');|

Exporting

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Excel Export | Not Applicable | **property:** allowExcelExport

var pivotGridObj = new
ej.pivotview.PivotView({
 allowExcelExport: true
});
pivotGridObj.appendTo('#PivotGrid');|

| Pdf Export | Not Applicable | **property:** allowPdfExport

var pivotGridObj = new
ej.pivotview.PivotView({
allowPdfExport: true
});
pivotGridObj.appendTo('#PivotGrid');|

Grid Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Set width for pivot table | Not Applicable | **property:** width

var pivotGridObj = new
ej.pivotview.PivotView({
width: '800'
});
pivotGridObj.appendTo('#PivotGrid');|

| Set height for pivot table | Not Applicable | **property:** height

var pivotGridObj = new
ej.pivotview.PivotView({
height: '400'
});
pivotGridObj.appendTo('#PivotGrid');|

| Set row height for pivot table | Not Applicable | **property:** rowHeight

var pivotGridObj = new
ej.pivotview.PivotView({
 gridSettings: { rowHeight: 60
}
});
pivotGridObj.appendTo('#PivotGrid');|

| Set column width for pivot table | Not Applicable | **property:** columnWidth

var pivotGridObj =
new ej.pivotview.PivotView({
 gridSettings: { columnWidth: 120
}
});
pivotGridObj.appendTo('#PivotGrid');|

| Drag and drop column headers in pivot table | Not Applicable | **property:** allowReordering

var
pivotGridObj = new ej.pivotview.PivotView({
 gridSettings: { allowReordering:
true}
});
pivotGridObj.appendTo('#PivotGrid');|

| Resizing the column headers in pivot table | **property:**
enableColumnResizing

\$("#PivotGrid").ejPivotGrid({
 enableColumnResizing:
true
}); | **property:** allowResizing

var pivotGridObj = new ej.pivotview.PivotView({

gridSettings: { allowResizing: true }
});
pivotGridObj.appendTo('#PivotGrid');|

| Wrap the cell content in pivot table | Not Applicable | **property:** allowTextWrap

var
pivotGridObj = new ej.pivotview.PivotView({
 gridSettings: { allowTextWrap: true
}
});
pivotGridObj.appendTo('#PivotGrid');|

| Display cell border in pivot table | Not Applicable | **property:** gridLines

var pivotGridObj = new ej.pivotview.PivotView({
 gridSettings: { gridLines:'Vertical'
});
pivotGridObj.appendTo('#PivotGrid');|

| Cell selection | **property:** enableCellSelection

\$("#PivotGrid").ejPivotGrid({
 enableCellSelection: true
 }); | **property:** allowSelection

var pivotGridObj = new ej.pivotview.PivotView({
gridSettings: { allowSelection: true,
 selectionSettings: {
 cellSelectionMode: 'Box',
 type: 'Multiple',
 mode: 'Cell' }
});
pivotGridObj.appendTo('#PivotGrid');|

| Display overflow cell content in pivot table | Not Applicable | **property:** clipMode

var pivotGridObj = new ej.pivotview.PivotView({
gridSettings: { clipMode: 'Clip'
});
pivotGridObj.appendTo('#PivotGrid');|

| Cell Editing | **property:** enableCellEditing

\$("#PivotGrid").ejPivotGrid({
 enableCellEditing:true
 }); | Not Applicable |

| Cell double click | **property:** enableCellDoubleClick

\$("#PivotGrid").ejPivotGrid({
 enableCellDoubleClick: true
 }); | Not Applicable |

| Cell context | **property:** enableCellContext

\$("#PivotGrid").ejPivotGrid({
enableCellContext: true
 }); | Not Applicable |

Drill Through

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Show/hide drill though feature | **property:** enableDrillThrough

\$("#PivotGrid").ejPivotGrid({
 enableDrillThrough: true
}); | **property:** allowDrillThrough

var pivotGridObj = new ej.pivotview.PivotView({
 allowDrillThrough: true
});
pivotGridObj.appendTo('#PivotGrid');|

| Event Triggers when cell clicked in pivot table widget | **event:** drillThrough

\$("#PivotGrid").ejPivotGrid({
drillThrough: function() {}
}); | **event:** drillThrough

var pivotGridObj = new ej.pivotview.PivotView({
 drillThrough: function() {}
});
pivotGridObj.appendTo('#PivotGrid');|

Cell Editing

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Edit settings | Not Applicable | **property:** editSettings

var pivotGridObj = new ej.pivotview.PivotView({
 editSettings: { }
});
pivotGridObj.appendTo('#PivotGrid');|

| Show/hide cell editing feature | **property:** enableCellEditing

\$("#PivotGrid").ejPivotGrid({
 enableCellEditing: true
}); | **property:** allowEditing

var pivotGridObj = new ej.pivotview.PivotView({
 editSettings: {
allowAdding: true, allowDeleting: true, allowEditing: true, mode: 'Normal'
}
});
pivotGridObj.appendTo('#PivotGrid');|

Hyperlink

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Hyperlink settings | **property:** hyperlinkSettings

\$("#PivotGrid").ejPivotGrid({
 hyperlinkSettings: { }
}); | **property:** hyperlinkSettings

var pivotGridObj = new ej.pivotview.PivotView({
 hyperlinkSettings: { }
});
pivotGridObj.appendTo('#PivotGrid'); |

| Show/hide hyperlink to all cells | Not Applicable | **property:** showHyperlink

var pivotGridObj = new ej.pivotview.PivotView({
 hyperlinkSettings: {
showHyperlink: true }
});
pivotGridObj.appendTo('#PivotGrid'); |

| Show/hide hyperlink to row headers | **property:** enableRowHeaderHyperlink

\$("#PivotGrid").ejPivotGrid({
 hyperlinkSettings: {
enableRowHeaderHyperlink: true }
}); | **property:** showRowHeaderHyperlink

var pivotGridObj = new ej.pivotview.PivotView({
 hyperlinkSettings: {
showRowHeaderHyperlink: true }
});
pivotGridObj.appendTo('#PivotGrid'); |

| Show/hide hyperlink to column headers | **property:** enableColumnHeaderHyperlink

\$("#PivotGrid").ejPivotGrid({
 hyperlinkSettings: {
enableColumnHeaderHyperlink: true }
}); | **property:** showColumnHeaderHyperlink

var pivotGridObj = new ej.pivotview.PivotView({
 hyperlinkSettings: {
showColumnHeaderHyperlink: true }
});
pivotGridObj.appendTo('#PivotGrid'); |

| Show/hide hyperlink to value cells | **property:** enableValueCellHyperlink

\$("#PivotGrid").ejPivotGrid({
 hyperlinkSettings: {
enableValueCellHyperlink: true }
}); | **property:** showValueCellHyperlink

var pivotGridObj = new ej.pivotview.PivotView({
 hyperlinkSettings: {
showValueCellHyperlink: true }
});
pivotGridObj.appendTo('#PivotGrid'); |

| Show/hide hyperlink to summary cells | **property:** enableSummaryCellHyperlink

\$("#PivotGrid").ejPivotGrid({
 hyperlinkSettings: {
enableSummaryCellHyperlink: true }
}); | **property:** showSummaryCellHyperlink

var pivotGridObj = new ej.pivotview.PivotView({
 hyperlinkSettings: {
showSummaryCellHyperlink: true }
});
pivotGridObj.appendTo('#PivotGrid'); |

| Show/hide hyperlink using specific conditions | Not Applicable | **property:** conditionalSettings

var pivotGridObj = new ej.pivotview.PivotView({
 hyperlinkSettings: {
conditionalSettings: [{
 measure: 'Units Sold', conditions: 'Between', value1: 150, value2: 200
}] }
});
pivotGridObj.appendTo('#PivotGrid'); |

| Show/hide hyperlink for row or column | Not Applicable | **property:** headerText

var pivotGridObj = new ej.pivotview.PivotView({
 hyperlinkSettings: {
headerText: 'FY 2015.Q1.Units Sold' }
});
pivotGridObj.appendTo('#PivotGrid'); |

| Event Triggers when row headers clicked in pivot table widget | **event:** rowHeaderHyperlinkClick

\$("#PivotGrid").ejPivotGrid({
rowHeaderHyperlinkClick: function() { }
}); | **event:** hyperlinkCellClick

var pivotGridObj = new ej.pivotview.PivotView({
 hyperlinkCellClick: function() { }
});
pivotGridObj.appendTo('#PivotGrid'); |

|Event Triggers when column headers clicked in pivot table widget| **event:**
 columnHeaderHyperlinkClick

\$("#PivotGrid").ejPivotGrid({
columnHeaderHyperlinkClick:
 function() {}
});| **event:** hyperlinkCellClick

var pivotGridObj = new
 ej.pivotview.PivotView({
 hyperlinkCellClick: function()
 {}
});
pivotGridObj.appendTo('#PivotGrid');|

|Event Triggers when value cells clicked in pivot table widget| **event:**
 valueCellHyperlinkClick

\$("#PivotGrid").ejPivotGrid({
valueCellHyperlinkClick: function()
 {}
});| **event:** hyperlinkCellClick

var pivotGridObj = new ej.pivotview.PivotView({

 hyperlinkCellClick: function() {}
});
pivotGridObj.appendTo('#PivotGrid');|

|Event Triggers when summary cells clicked in pivot table widget| **event:**
 summaryCellHyperlinkClick

\$("#PivotGrid").ejPivotGrid({
summaryCellHyperlinkClick:
 function() {}
});| **event:** hyperlinkCellClick

var pivotGridObj = new
 ej.pivotview.PivotView({
 hyperlinkCellClick: function()
 {}
});
pivotGridObj.appendTo('#PivotGrid');|

Defer Layout Update

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Show/hide defer layout update| **property:**
 enableDeferUpdate

\$("#PivotGrid").ejPivotGrid({
 enableDeferUpdate:
 true
});| **property:** allowDeferLayoutUpdate

var pivotGridObj = new
 ej.pivotview.PivotView({
 allowDeferLayoutUpdate:
 true
});
pivotGridObj.appendTo('#PivotGrid');|

Accessibility

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Localization| **property:** locale

\$("#PivotGrid").ejPivotGrid({
locale: 'es-ES'

 });| **property:** locale

var pivotGridObj = new ej.pivotview.PivotView({
locale: 'es-
 ES'
});
pivotGridObj.appendTo('#PivotGrid');|

|Right to left| **property:** enableRTL

\$("#PivotGrid").ejPivotGrid({
 enableRTL:
 true
});| **property:** enableRtl

var pivotGridObj = new ej.pivotview.PivotView({

 enableRtl: true
});
pivotGridObj.appendTo('#PivotGrid');|

Common

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Adding custom class to wrapper element| **property:**
 cssClass

\$("#PivotGrid").ejPivotGrid({
 cssClass: 'custom-class'
 });| **property:**
 cssClass

var pivotGridObj = new ej.pivotview.PivotView({
cssClass: 'custom-
 class'
});
pivotGridObj.appendTo('#PivotGrid');|

|Keeping the model values in cookies|Not Applicable| **property:** enablePersistence

var
 pivotGridObj = new ej.pivotview.PivotView({
 enablePersistence:
 true});
pivotGridObj.appendTo('#PivotGrid');|

| Event triggers when control initializing | **event:** load

\$("#PivotGrid").ejPivotGrid({
 load: function() {}
}); | **event:** load

var pivotGridObj = new ej.pivotview.PivotView({
 load: function() {}
});
pivotGridObj.appendTo('#PivotGrid'); |

| Event Triggers before the pivot engine starts | **event:** beforePivotEnginePopulate

\$("#PivotGrid").ejPivotGrid({
 beforePivotEnginePopulate: function() {}
 }); | **event:** enginePopulating

var pivotGridObj = new ej.pivotview.PivotView({
 enginePopulating: function() {}
});
pivotGridObj.appendTo('#PivotGrid'); |

| Event Triggers after the pivot engine populated | **event:** afterPivotEnginePopulate

\$("#PivotGrid").ejPivotGrid({
 afterPivotEnginePopulate: function() {}
 }); | **event:** enginePopulated

var pivotGridObj = new ej.pivotview.PivotView({
 enginePopulated: function() {}
});
pivotGridObj.appendTo('#PivotGrid'); |

| Event Triggers after the control populated with data source | **event:** renderSuccess

\$("#PivotGrid").ejPivotGrid({
 renderSuccess: function() {}
 }); | **event:** dataBound

var pivotGridObj = new ej.pivotview.PivotView({
 dataBound: function() {}
});
pivotGridObj.appendTo('#PivotGrid'); |

| Event Triggers after the control created | Not Applicable | **event:** created

var pivotGridObj = new ej.pivotview.PivotView({
 created: function() {}
});
pivotGridObj.appendTo('#PivotGrid'); |

| Event Triggers when destroy the control | Not Applicable | **event:** destroyed

var pivotGridObj = new ej.pivotview.PivotView({
 destroyed: function() {}
});
pivotGridObj.appendTo('#PivotGrid'); |

| Event Triggers the cell clicked in pivot table widget | **event:** cellClick

\$("#PivotGrid").ejPivotGrid({
 cellClick: function() {}
}); | **event:** cellClick

var pivotGridObj = new ej.pivotview.PivotView({
 cellClick: function() {}
});
pivotGridObj.appendTo('#PivotGrid'); |

How To

<!-- markdownlint-disable MD009 -->

Switching older themes style in EJ2 JavaScript Pivotview control

From Volume 1, 2020 onwards Syncfusion has revised the theming and layout of the Pivot Table. So, to inherit the older theme style and layout please do the necessary changes in CSS and pivot table height.

CSS Selectors

In current theme, the cells can be differentiated by their background colors. To avoid it, you need to override its background colors via simple CSS coding within your application. The below CSS selectors allow to achieve the same for material, fabric, bootstrap and bootstrap v4 themes.

```
`html
```

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
```

```
<!-- Codes here... -->
```



```

<style>
.e-pivotview .e-rowsheader,
.e-pivotview .e-columnsheader,
.e-pivotview .e-gtot,
.e-pivotview .e-content,
.e-pivotview .e-gridheader,
.e-pivotview .e-headercell {
background-color:#fff !important;
}
</style>
</head>
<body>
</body>
</html>
`

```

Meanwhile for high contrast theme, we need to set the following CSS.

```

`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<!-- Codes here... -->
<style>
.e-pivotview .e-rowsheader,
.e-pivotview .e-columnsheader,
.e-pivotview .e-gtot,
.e-pivotview .e-content,
.e-pivotview .e-gridheader,
.e-pivotview .e-headercell {
background-color:#000 !important;
}
</style>
</head>
<body>

```

</body>

</html>

,

Adjusting Row Height

In current theme, to make the component compact we have reduced the height of each pivot table rows. But user can reset the height of the pivot table using the [rowHeight](#) property in [gridSettings](#). In older theme, the property was set to 36 pixels for desktop layout and 48 pixels for mobile layout. So reset the [rowHeight](#) accordingly to visualize the older theme style.

In the below code sample, we replicate the older theme style.

INDEX.TS

```
import { PivotView, IDataset } from '@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
let pivotTableObj: PivotView = new PivotView({
  dataSourceSettings: {
    dataSource: pivotData as IDataset[],
    expandAll: false,
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  gridSettings: {
    rowHeight: 36
  }
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<style>
    .e-pivotview .e-rowsheader,
    .e-pivotview .e-columnsheader,
    .e-pivotview .e-gtot,
    .e-pivotview .e-content,
    .e-pivotview .e-gridheader,
    .e-pivotview .e-headercell {
        background-color:#fff !important;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize number date and time values in EJ2 JavaScript Pivotview control

You can format the number, date, and time values for each field using `formatSettings` option under `dataSourceSettings`. It can be configured through code behind, during initial rendering.

Number formatting

For numbers, the formatting settings required to apply through code behind are:

- `name`: It allows to set the field name.
- `format`: It allows to set the format of the respective field.

Also, you can customize the applied number format by setting the [NumberFormatOptions](#) options in `formatSettings` itself.

INDEX.TS

```
import { PivotView, IDataset } from '@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataset[],
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        formatSettings: [{ name: 'Amount', format: 'C2', useGrouping: false,
            minimumSignificantDigits: 1, maximumSignificantDigits: 3 }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: []
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">
```

```

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Date and time formatting

For date and time, the formatting settings required to apply through code behind are:

- **name:** It allows to set the field name.
- **format:** It allows to set the format of the respective field.
- **type:** It allows to set the type of format to be used for the respective field.

Also, you can customize the applied date format by setting [DateFormatOptions](#) options in `formatSettings` itself.

INDEX.TS

```

import { PivotView, IDataset } from '@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataset[],
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        formatSettings: [{ name: 'Year', format: 'dd/MM/yyyy-hh:mm', type:
'date' }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: []
    },

```

```

        height: 350
    });
    pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations of date formatting

As per Firefox and Edge browsers standards, most of the date and time formats used in data source aren't supported. For example: Apr-2000, Apr-01-2000, 01-03-2000, 2000-Apr-01 etc... are not supported. Meanwhile [ISO formats](#) will be supported across all browsers.

Customize the icons for pivot table in EJ2 JavaScript Pivotview control

You can customize the pivot button icons in the pivot table by overriding the class **.pivot-button** with a custom property content as mentioned below.

```
`ts
```

```
PivotView_PivotFieldList.e-icons.e-toggle-field-list::before {
content: '\e337';
```

```
}
```

```
`
```

In the below sample, pivot table is rendered with a customized pivot button icons.

INDEX.TS

```

import { PivotView, IDataset, FieldList } from '@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
PivotView.Inject(FieldList);
let pivotTableObj: PivotView = new PivotView({
  dataSourceSettings: {
    dataSource: pivotData as IDataset[],
    expandAll: false,
    enableSorting: true,
    allowLabelFilter: true,
    allowValueFilter: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  showFieldList: true,
  height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Pivot Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="PivotFieldList"></div>
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Configure data grid options on editing mode in EJ2 JavaScript Pivotview control

You can access the data grid options such as sort, group, filter on editing mode using the **beginDrillThrough** event in the pivot table. The event occurs in every value cell on double click and provides the data grid information before display the drill through grid pop-up.

Grid features are segregated into individual feature-wise modules. For example, to use sorting feature, you should inject **Sort** using the **Grid.Inject(Sort)** section.

INDEX.TS

```

import { PivotView, IDataSet, BeginDrillThroughEventArgs } from
 '@syncfusion/ej2-pivotview';
import { Grid, Sort, Filter, Group } from '@syncfusion/ej2-grids';
import { pivotData } from './datasource.ts';
Grid.Inject(Sort, Filter, Group);
let pivotTableObj: PivotView = new PivotView({
  dataSourceSettings: {
    dataSource: pivotData as IDataSet[],
    expandAll: false,
    enableSorting: true,
    drilledMembers: [{ name: 'Country', items: ['France'] }],
    columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
    values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    filters: []
  },
  height: 350,
  editSettings: { allowAdding: true, allowDeleting: true, allowEditing:
true, mode: 'Normal' },
  beginDrillThrough: (args: BeginDrillThroughEventArgs) => {
    if (args.gridObj) {
      let gridObj: Grid = args.gridObj;
      gridObj.allowGrouping = true;
      gridObj.allowSorting = true;
      gridObj.allowFiltering = true;
      gridObj.filterSettings = { type: 'CheckBox' };
    }
  }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Refresh the field list in EJ2 JavaScript Pivotview control

You can refresh pivot table and field list with new data source dynamically.

INDEX.TS

```

import { PivotView, FieldList } from '@syncfusion/ej2-pivotview';
import { Button } from '@syncfusion/ej2-buttons';
PivotView.Inject(FieldList);
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: [
            { 'Sold': 31, 'Amount': 52824, 'Country': 'France', 'Products':
'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q1' },
            { 'Sold': 51, 'Amount': 86904, 'Country': 'France', 'Products':
'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q2' },
            { 'Sold': 90, 'Amount': 153360, 'Country': 'France', 'Products':
'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q3' },

```

```

        { 'Sold': 25, 'Amount': 42600, 'Country': 'France', 'Products':
'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q4' },
        { 'Sold': 27, 'Amount': 46008, 'Country': 'France', 'Products':
'Mountain Bikes', 'Year': 'FY 2016', 'Quarter': 'Q1' }]],
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    showFieldList: true,
    height: 350
});
pivotTableObj.appendTo('#PivotTable');
let btn: Button = new Button({ isPrimary: true });
btn.appendTo('#Refresh');
document.getElementById('Refresh').addEventListener('click', () => {
    pivotTableObj.engineModule.fieldList = {};
    pivotTableObj.dataSourceSettings.dataSource = [
        { 'Sold': 31, 'Amount': 52824, 'Country': 'France', 'Year': 'FY
2015' },
        { 'Sold': 51, 'Amount': 86904, 'Country': 'France', 'Year': 'FY
2015' },
        { 'Sold': 90, 'Amount': 153360, 'Country': 'France', 'Year': 'FY
2015' },
        { 'Sold': 25, 'Amount': 42600, 'Country': 'France', 'Year': 'FY
2015' },
        { 'Sold': 27, 'Amount': 46008, 'Country': 'France', 'Year': 'FY
2016' }]];
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Refresh">Refresh</div>
        <div id="PivotTable"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hide empty headers in EJ2 JavaScript Pivotview control

If the raw data for a particular field is not defined, it will be shown as 'Undefined' in the pivot table headers. You can hide those headers by setting the [showHeaderWhenEmpty](#) property to **false** in the pivot table.

For example, if the raw data for the field 'Country' is defined as **"United Kingdom"** and **"State"** is not defined means, it will be shown as **"United Kingdom >> Undefined"** in the header section. Here, you can hide those 'Undefined' header using the [showHeaderWhenEmpty](#) property.

By default, this property is set as **true**.

INDEX.TS

```

import { PivotView, IDataSet } from '@syncfusion/ej2-pivotview';
import { pivotNullData } from './datasource.ts';
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotNullData as IDataSet[],
        expandAll: false,
        rows: [{ name: 'Country' }, { name: 'State' }],
        columns: [{ name: 'Product', showNoDataItems: true }, { name: 'Date'
    }],
        values: [{ name: 'Amount' }, { name: 'Quantity' }],
        showHeaderWhenEmpty: false
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

Customizing loading indicator in EJ2 JavaScript Pivotview control

You can customize the appearance of the loading indicator in the pivot table by using the [spinnerTemplate](#) property. This property accepts an HTML string which can be used for appearance customization.

You can also disable the loading indicator by setting [spinnerTemplate](#) to empty string.

INDEX.TS

```
import { PivotView, IDataset } from '@syncfusion/ej2-pivotview';
import { DataManager, WebApiAdaptor, Query, ReturnOption } from
 '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://bi.syncfusion.com/northwindservice/api/orders',
    adaptor: new WebApiAdaptor,
    crossDomain: true
});
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: data,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
    },
    height: 350,
    spinnerTemplate: '<i class="fa fa-cog fa-spin fa-3x fa-fw"></i>'
});
pivotTableObj.appendTo('#PivotTable');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing the pivotview component minimum height in EJ2 JavaScript Pivotview control

The **minHeight** property allows you to change the minimum height for the pivot table control. For the pivot table control, the default minimum height is **300px**.

INDEX.TS

```

import { PivotView, IDataset } from '@syncfusion/ej2-pivotview';
import { Button } from '@syncfusion/ej2-buttons';
import { pivotData } from './datasource.ts';
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        values: [{ name: 'Amount', caption: 'Sold Amount' },
            { name: 'Sold', caption: 'Units Sold' }],
    }
});

```

```

        filters: [],
    },
    height: 200
});
pivotTableObj.minHeight = 200;
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>
</body>
</html>

```



```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Chart based on pivot table selection in EJ2 JavaScript Pivotview control

The cell selection support is enabled using the [allowSelection](#) property and its type and mode are configured using the [selectionSettings](#) property. The [cellSelected](#) event gets fired on every selection operation performed in the pivot table. This event returns the selected cell informations, like row header name, column header name, measure name, and value. Based on this information, the [chart](#) control will be plotted.

INDEX.TS

```

import { Chart, Category, Legend, Tooltip, ColumnSeries, LineSeries,
SeriesModel } from '@syncfusion/ej2-charts';
import { PivotView, IDataset, PivotCellSelectedEventArgs } from
 '@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
let onInit: boolean = true;
let measureList: { [key: string]: string } = {};
let chart: Chart;
let selectedCells: CellSelectedObject[];
let chartSeries: SeriesModel[];
let pivotObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataset[],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: []
    },
    width: '100%',
    height: 290,
    dataBound: () => {
        if (onInit) {
            for (let value of pivotObj.dataSourceSettings.values) {
                measureList[value.name] = value.caption || value.name;
            }
            pivotObj.grid.selectionModule.selectCellsByRange({ cellIndex: 1,
rowIndex: 1 }, { cellIndex: 3, rowIndex: 3 });
        }
    },
    gridSettings: {
        allowSelection: true,
        selectionSettings: { mode: 'Both', type: 'Multiple',
cellSelectionMode: 'Box' }
    },
    cellSelected: (args: PivotCellSelectedEventArgs) => {
        selectedCells = args.selectedCellsInfo;
        if (selectedCells && selectedCells.length > 0) {

```

```

        chartSeries = frameChartSeries();
        chartUpdate();
    }
}
});
pivotObj.appendTo('#PivotTable');
function frameChartSeries(): SeriesModel[] {
    let columnGroupObject: { [key: string]: { x: string, y: number }[] } =
    {};
    for (let cell of selectedCells) {
        if (cell.measure !== '') {
            let columnSeries = (pivotObj.dataSourceSettings.values.length > 1 &&
measureList[cell.measure]) ?
            (cell.columnHeaders.toString() + ' ~ ' + measureList[cell.measure])
: cell.columnHeaders.toString();
            if (columnGroupObject[columnSeries]) {
                columnGroupObject[columnSeries].push({ x: cell.rowHeaders == '' ?
'Grand Total' : cell.rowHeaders.toString(), y: Number(cell.value) });
            } else {
                columnGroupObject[columnSeries] = [{ x: cell.rowHeaders == '' ?
'Grand Total' : cell.rowHeaders.toString(), y: Number(cell.value) }];
            }
        }
    }
    let columnKeys: string[] = Object.keys(columnGroupObject);
    let chartSeries: SeriesModel[] = [];
    for (let key of columnKeys) {
        chartSeries.push({
            dataSource: columnGroupObject[key],
            xName: 'x',
            yName: 'y',
            type: 'Column',
            name: key
        });
    }
    return chartSeries;
}
function chartUpdate() {
    if (onInit) {
        onInit = false;
        Chart.Inject(ColumnSeries, LineSeries, Legend, Tooltip, Category);
        chart = new Chart({
            title: 'Sales Analysis',
            legendSettings: {
                visible: true
            },
            tooltip: {
                enable: true
            },
            primaryYAxis: {
                title: pivotObj.dataSourceSettings.values.map(function
(args) { return args.caption || args.name }).join(' ~ '),
            },
            primaryXAxis: {
                valueType: 'Category',
                title: pivotObj.dataSourceSettings.rows.map(function (args)
{ return args.caption || args.name }).join(' ~ '),

```

```

        labelIntersectAction: 'Rotate45'
    },
    series: chartSeries,
}, '#Chart');
} else {
    chart.series = chartSeries;
    chart.primaryXAxis.title =
pivotObj.dataSourceSettings.rows.map(function (args) { return args.caption
|| args.name }).join(' ~ ');
    chart.primaryYAxis.title =
pivotObj.dataSourceSettings.values.map(function (args) { return args.caption
|| args.name }).join(' ~ ');
    chart.refresh();
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div id="PivotTable"></div>
        <br>
        <div id="Chart"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drill through grid cell edit type in EJ2 JavaScript Pivotview control

Using the [drillThrough](#) event in the pivot table, you can define the edit type of a particular column in the grid present inside the drill-through dialog. To do so, check the column name in the [drillThrough](#) event and then specify the edit type of that column using the [gridColumns.editType](#) event argument.

The [gridColumns.editType](#) property must be set based on the column's data type. For example, the string data type will not be applicable for the numeric text box edit type.

- [NumericTextBox](#) control for integer, double, and decimal data types.
- [TextBox](#) control for string data type.
- [DropDownList](#) control to show all unique values related to that field.
- [CheckBox](#) control for boolean data type.
- [DatePicker](#) control for date data type.
- [DateTimePicker](#) control for date time data type.

In the below example, the data type of the **Country** column is set to **DropDownList**.

INDEX.TS

```

import { PivotView, IDataset } from '@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataset[],
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    height: 350,
    drillThrough(args) {
        for (var i = 0; i < args.gridColumns.length; i++) {
            if (args.gridColumns[i].field === 'Country') {
                args.gridColumns[i].editType = 'dropdownedit';
                //args.gridColumns[i].editType = 'numericedit';
            }
        }
    }
});

```

```

        //args.gridColumns[i].editType = 'textedit';
        //args.gridColumns[i].editType = 'booleanedit';
        //args.gridColumns[i].editType = 'datepickeredit';
        //args.gridColumns[i].editType = 'datetimepickeredit';
    }
}
},
editSettings: { allowAdding: true, allowDeleting: true, allowEditing:
true, mode: 'Normal' }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show field list when pivot table empty in EJ2 JavaScript Pivotview control

When there are no fields in a pivot table's row, column, value, and filter axes, a field list can still be displayed. To do so, use the [dataBound](#) event and call the [onShowFieldList](#) method as shown below.

INDEX.TS

```

import { PivotView, IDataset, FieldList } from '@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
PivotView.Inject(FieldList);
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataset[],
    },
    showFieldList: true,
    dataBound: (): void => {
        if (pivotTableObj && pivotTableObj.dataSourceSettings.values.length
        === 0) {
            (pivotTableObj.pivotFieldListModule.dialogRenderer as
            any).onShowFieldList();
        }
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Apply custom style to pivot cells

The [queryCellInfo](#) event in [gridSettings](#) can be used to apply custom style to row and value cells, and the [headerCellInfo](#) event in [gridSettings](#) can be used to apply custom styles to column cells.

In the following example, a custom style has been applied to the column header **“Sold Amount”** under **“FY 2015”** via the [headerCellInfo](#) event and to the row header **“Germany”** and its aggregated value via the [queryCellInfo](#) event by adding the **“e-custom”** class to the cell element.

INDEX.TS

```

import { PivotView, IDataset } from '@syncfusion/ej2-pivotview';
import { QueryCellInfoEventArgs, HeaderCellInfoEventArgs } from
 '@syncfusion/ej2-grids';
import { pivotData } from './datasource.ts';
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {

```

```

        dataSource: pivotData as IDataset[],
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    gridSettings: {
        queryCellInfo: function (args: QueryCellInfoEventArgs) {
            let colIndex: number = Number(args.cell.getAttribute('data-
colindex'));
            let cells: QueryCellInfoEventArgs = args.data[colIndex] ?
args.data[colIndex] : {};
            // Here by using 'actualText' option, a custom class can be
added to the specific row header and its value to apply custom style.
            if (cells.actualText === 'Germany') {
                args.cell.classList.add('e-custom');
            } else if (cells.actualText === 'Amount' &&
cells.columnHeaders === 'FY 2016' && cells.rowHeaders ===
'Germany') {
                args.cell.classList.add('e-custom');
            }
        },
        headerCellInfo: function (args: HeaderCellInfoEventArgs) {
            let customAttributes: HeaderCellInfoEventArgs =
args.cell.column.customAttributes;
            // Here custom class can be added to the specific column header
by using unique level name, to apply custom style.
            if (args.node.classList.contains('e-columnsheader') &&
customAttributes &&
customAttributes.cell.valueSort.levelName === 'FY 2016.Sold
Amount') {
                args.node.classList.add('e-custom');
            }
        }
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<style>
    .e-custom {
        font-family: 'Courier New', Courier, monospace;
        font-size: 12px !important;
        background-color: pink !important;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: The **dot(.)** character in **FY 2016.Sold Amount** is used by default to identify the header levels in the pivot table's row and column. It can be changed by setting the [headerDelimiter](#) in the [valueSortSettings](#) property to any other delimiter instead of the default separator.

Show tooltip for row and column headers

You can create and display the tooltip for each row and column header(s) in the pivot table by using an external tooltip component via the [dataBound](#) event.

INDEX.TS

```
import { PivotView, IDataSet, IAxisSet } from '@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
import { Tooltip } from '@syncfusion/ej2-popups';
let headerTooltip: Tooltip;
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataSet[],
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    dataBound: (): void => {
        if (!headerTooltip) {
            headerTooltip = new Tooltip({
                target: 'td.e-rowsheader,th.e-columnsheader', beforeRender:
beforeRender
            });
            headerTooltip.appendTo(pivotTableObj.element);
        }
    },
    height: 350
});
pivotTableObj.appendTo('#PivotTable');
// Method used to customize the tooltip content while hovering each row and
column headers in the pivot table.
function beforeRender(args: any) {
    if (args.target.parentElement.querySelector('.e-rowsheader')) {
        // Here you can set custom content for row header(s) tooltip from
its cell information.
        let index: number =
Number(args.target.getAttributeNode('index').value);
        let colIndex: number = Number(args.target.getAttributeNode('data-
colindex').value);
        let cell: IAxisSet =
pivotTableObj.engineModule.pivotValues[index][colIndex];
        let valueText: any = cell.valueSort ? cell.valueSort : '';
        if (cell.formattedText !== 'Grand Total') {
            this.content =
                '<div>' +
                'FieldName: ' +
                valueText.axis +
```

```

        '</br>' +
        'Text: ' +
        cell.formattedText +
        '</div>';
    } else {
        this.content =
        '<div>' +
        'FieldName: ' +
        valueText.uniqueName +
        '</br>' +
        'Text: ' +
        cell.formattedText +
        '</div>';
    }
} else {
    // Here you can set custom content for column header(s) tooltip from
    // its cell information.
    if (args.target.querySelector('.e-cellvalue')) {
        this.content = args.target.querySelector('.e-
cellvalue').innerText;
    } else if (args.target.querySelector('.e-headertext')) {
        this.content = args.target.querySelector('.e-
headertext').innerText;
    } else if (args.target.querySelector('.e-stackedheadercelldiv')) {
        this.content = args.target.querySelector('.e-
stackedheadercelldiv').innerText;
    } else {
        this.content = '';
    }
}
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hide specific columns in pivot table

By using the [columnRender](#) event in the [gridSettings](#), you can hide specific column(s) in the pivot table. In the example below, the “Units Sold” column under “FY 2016” is hidden by setting its **visible** property to **false** via the [columnRender](#) event.

Note: The **dot(.)** character in **FY 2016.Units Sold** is used by default to identify the header levels in the pivot table's row and column. It can be changed by setting the [headerDelimiter](#) in the [valueSortSettings](#) property to any other delimiter instead of the default separator.

INDEX.TS

```

import { PivotView, IDataset, ColumnRenderEventArgs } from '@syncfusion/ej2-
pivotview';
import { pivotData } from './datasource.ts';
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataset[],
        expandAll: false,
        enableSorting: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],

```

```

        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
    },
    gridSettings: {
        columnRender: function (args: ColumnRenderEventArgs) {
            // Specific column(s) can be hidden by checking their level name
            and setting its visible property accordingly.
            for (let i = 1; i < args.columns.length; i++) {
                if (args.stackedColumns[i].customAttributes &&
args.stackedColumns[i].customAttributes.cell.valueSort.levelName === 'FY
2016.Units Sold') {
                    args.stackedColumns[i].visible = false;
                }
            }
        },
        height: 350
    });
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

```

```

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export table and chart into the same document using toolbar

Even if the [displayOption.view](#) property is set to **Both** in the pivot table, you can only export either the table or the chart to the PDF document based on the current value set in the [displayOption.primary](#) property. But, to export both the table and the chart to the same PDF document, use the [pdfExport](#) method during the [actionBegin](#) event invoke.

In the following example, the built-in export action can be restricted by setting the [args.cancel](#) option to **true** in the [actionBegin](#) event, and both the table and the chart can be exported by calling the [pdfExport](#) method and setting the `exportBothTableAndChart` argument to **true**.

INDEX.TS

```

import {
    PivotView, FieldList, CalculatedField, Toolbar, IDataset, PDFExport,
    PivotActionBeginEventArgs
} from '@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
PivotView.Inject(FieldList, CalculatedField, Toolbar, PDFExport);
let pivotTableObj: PivotView = new PivotView({
    dataSourceSettings: {
        dataSource: pivotData as IDataset[],
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        sortSettings: [{ name: 'Country', order: 'Descending' }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],

```

```

        filters: []
    },
    height: 350,
    actionBegin: function (args: PivotActionBeginEventArgs) {
        if (args.actionName == 'PDF export') {
            args.cancel = true;
            pivotTableObj.pdfExport({}, false, null, false, true);
        }
    },
    toolbar: ['Grid', 'Chart', 'Export', 'FieldList'],
    allowPdfExport: true,
    showToolbar: true,
    displayOption: { view: 'Both' },
    allowCalculatedField: true,
    showFieldList: true,
    gridSettings: { columnWidth: 140 }
});
pivotTableObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD009 -->

Add custom aggregation type to the menu in EJ2 JavaScript Pivotview control

By using the [dataBound](#) event, you can add your own custom aggregate type(s) to the pivot table's aggregate menu.

In the following example, we have added the aggregation types **CustomAggregateType 1** and **CustomAggregateType 2** to the aggregate menu. The calculation for those aggregated types can be done using the [aggregateCellInfo](#) event.

INDEX.TS

```

import { PivotView, FieldList, IDataset, AggregateTypes } from
 '@syncfusion/ej2-pivotview';
import { L10n } from '@syncfusion/ej2-base';
import { pivotData } from './datasource.ts';
PivotView.Inject(FieldList);
L10n.load({
    'en-US': {
        pivotview: {
            CustomAggregateType1: 'Custom Aggregate Type 1',
            CustomAggregateType2: 'Custom Aggregate Type 2',
        },
        pivotfieldlist: {
            CustomAggregateType1: 'Custom Aggregate Type 1',
            CustomAggregateType2: 'Custom Aggregate Type 2',
        }
    }
});
const SummaryType: string[] = [
    'Sum',
    'Count',
    'DistinctCount',
    'Avg',
    'CustomAggregateType1',
    'CustomAggregateType2'
]

```



```

];
let pivotObj: PivotView = new PivotView({
  dataSourceSettings: {
    expandAll: false,
    dataSource: pivotData as IDataset[],
    columns: [{ name: 'Year' }, { name: 'Quarter' }],
    values: [{ name: 'Sold' }, { name: 'Amount' }],
    rows: [{ name: 'Country' }, { name: 'Products' }],
    formatSettings: [{ name: 'Amount', format: 'C0' }],
    subTotalsPosition: 'Bottom'
  },
  width: '100%',
  height: 300,
  showFieldList: true,
  dataBound: function () {
    pivotObj.getAllSummaryType = function () {
      return SummaryType as AggregateTypes[];
    };
    pivotObj.pivotFieldListModule.aggregateTypes = SummaryType as
AggregateTypes[];
    pivotObj.pivotFieldListModule.getAllSummaryType = function () {
      return SummaryType as AggregateTypes[];
    };
  },
  aggregateCellInfo(args) {
    if (args.aggregateType === 'CustomAggregateType1') {
      args.value = args.value * 100;
    }
    if (args.aggregateType === 'CustomAggregateType2') {
      args.value = args.value / 100;
    }
  }
});
pivotObj.appendTo('#PivotTable');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD009 -->

Convert complex JSON to flat JSON and assign it to the pivot table in EJ2 JavaScript Pivotview control

By default, flat JSON can only bind to the pivot table. However, you can connect complex JSON to the pivot table by converting it to flat JSON via code-behind and binding it to the pivot table using the [dataSource](#) property in the [load](#) event.

In the following example, the **complexToFlatJson()** method is used to convert complex JSON to flat JSON and bind it to the pivot table using the [dataSource](#) property, then modifying the field names in the [rows](#) and [columns](#) based on the converted flat JSON under [dataSourceSettings](#) in the [load](#) event.

INDEX.TS

```

import { PivotView, FieldList, LoadEventArgs } from '@syncfusion/ej2-
pivotview';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(false);
PivotView.Inject(FieldList);
let parentProp: any = {};

```

```

let dataSource: Object[][];
let pivotGridObj: PivotView = new PivotView({
    dataSourceSettings: {
        expandAll: true,
        enableSorting: true,
        dataSource: data() as any,
        columns: [{ name: 'OrderDetails' }],
        values: [{ name: 'Freight', caption: 'Units Sold' }],
        rows: [{ name: 'ShipDetails' }],
        valueSortSettings: { headerDelimiter: ' - ' },
        formatSettings: [{ name: 'Amount', format: 'C0' }]
    },
    load: function (args: LoadEventArgs) {
        dataSource =
JSON.parse(JSON.stringify(args.dataSourceSettings.dataSource));
        args.dataSourceSettings.dataSource =
complexToFlatJson(dataSource);
        let rows: any = [];
        for (let i: number = 0; i < args.dataSourceSettings.rows.length;
i++) {
            if (args.dataSourceSettings.rows[i].name in parentProp) {
                rows =
rows.concat(parentProp[args.dataSourceSettings.rows[i].name]);
            } else {
                rows.push(args.dataSourceSettings.rows[i]);
            }
        }
        args.dataSourceSettings.rows = rows;
        let columns: any = [];
        for (let i: number = 0; i <
args.dataSourceSettings.columns.length; i++) {
            if (args.dataSourceSettings.columns[i].name in parentProp) {
                columns = columns.concat(
                    parentProp[args.dataSourceSettings.columns[i].name]
                );
            } else {
                columns.push(args.dataSourceSettings.columns[i]);
            }
        }
        args.dataSourceSettings.columns = columns;
    },
    showFieldList: true,
    width: '100%',
    height: 500,
    gridSettings: { columnWidth: 140 },
});
pivotGridObj.appendTo('#PivotTable');
function complexToFlatJson(data: Object[][]): {
    let flatArray: any = [];
    let flatObject: any = {};
    for (let index = 0; index < data.length; index++) {
        for (let prop in data[index]) {
            let value: Object = data[index][prop];
            if (Array.isArray(value)) {
                for (let i: number = 0; i < value.length; i++) {
                    let childProp: any = [];
                    for (let inProp in value[i]) {

```

```

        flatObject[inProp] = value[i][inProp];
        let object = {
            name: inProp,
        };
        childProp.push(object);
    }
    parentProp[prop] = childProp;
}
} else {
    flatObject[prop] = value;
}
}
flatArray.push(flatObject);
flatObject = {};
}
return flatArray;
}
function data() {
    return [
        {
            CustomerID: 'VINET',
            Freight: 32.38,
            OrderDetails: [
                {
                    OrderID: 10248,
                    OrderDate: '1996-07-04T10:10:00.000Z',
                }
            ],
            ShipDetails: [
                {
                    ShipName: 'Vins et alcools Chevalier',
                    ShipAddress: '59 rue de l'Abbaye',
                    ShipCity: 'Reims',
                    ShipRegion: null,
                    ShipCountry: 'France',
                    ShippedDate: '1996-07-16T12:20:00.000Z',
                }
            ]
        },
        {
            CustomerID: 'GALED',
            Freight: 10.14,
            OrderDetails: [
                {
                    OrderID: 10366,
                    OrderDate: '1996-11-28T00:00:00.000Z',
                }
            ],
            ShipDetails: [
                {
                    ShippedDate: '1996-12-30T00:00:00.000Z',
                    ShipName: 'Galería del gastronómo',
                    ShipAddress: 'Rambla de Cataluña, 23',
                    ShipCity: 'Barcelona',
                    ShipRegion: null,
                    ShipCountry: 'Spain',
                }
            ]
        }
    ]
}

```

```

    ],
    {
        CustomerID: 'VAFFE',
        Freight: 13.55,
        OrderDetails: [
            {
                OrderID: 10367,
                OrderDate: '1996-12-02T00:00:00.000Z',
            }
        ],
        ShipDetails: [
            {
                ShippedDate: '1996-12-30T00:00:00.000Z',
                ShipName: 'Vaffeljernet',
                ShipAddress: 'Smagsloget 45',
                ShipCity: 'Århus',
                ShipRegion: null,
                ShipCountry: 'Denmark',
            }
        ]
    },
    {
        CustomerID: 'ERNSH',
        Freight: 101.95,
        OrderDetails: [
            {
                OrderID: 10368,
                OrderDate: '1996-11-29T00:00:00.000Z',
            }
        ],
        ShipDetails: [
            {
                ShippedDate: '1996-12-30T00:00:00.000Z',
                ShipName: 'Ernst Handel',
                ShipAddress: 'Kirchgasse 6',
                ShipCity: 'Graz',
                ShipRegion: null,
                ShipCountry: 'Austria',
            }
        ]
    },
    {
        CustomerID: 'SPLIR',
        Freight: 195.68,
        OrderDetails: [
            {
                OrderID: 10369,
                OrderDate: '1996-11-28T00:00:00.000Z',
            }
        ],
        ShipDetails: [
            {
                ShippedDate: '1996-12-30T00:00:00.000Z',
                ShipName: 'Split Rail Beer & Ale',
                ShipAddress: 'P.O. Box 555',
                ShipCity: 'Lander',
            }
        ]
    }
]

```

```

        ShipRegion: 'WY',
        ShipCountry: 'USA',
    },
]
}
];
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

  <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <div id="PivotTable"></div>
    </div>
  </div>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD009 -->

Load desired report from the report list as default in EJ2 JavaScript Pivotview control

By default, the pivot table is displayed with the report bound at the code-behind. To load a desired report from the previously saved report collection during initial rendering, set the desired report name in the [dataBound](#) event, along with the additional report-based customization code shown below.

INDEX.TS

```

import {
    PivotView, FieldList, CalculatedField, Toolbar, RemoveReportArgs,
    ToolbarArgs,
    ConditionalFormatting, IDataset, RenameReportArgs, SaveReportArgs,
    FetchReportArgs, LoadReportArgs, NumberFormatting
} from '@syncfusion/ej2-pivotview';
import { getInstance, select } from '@syncfusion/ej2-base';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { pivotData } from './datasource.ts';
PivotView.Inject(FieldList, CalculatedField, Toolbar, ConditionalFormatting,
    NumberFormatting);
let isInitial: boolean = true;
let pivotObj: PivotView = new PivotView({
    dataSourceSettings: {
        expandAll: false,
        dataSource: pivotData as IDataset[],
        columns: [{ name: 'Year' }, { name: 'Quarter' }],
        values: [{ name: 'Sold' }, { name: 'Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
    },
    width: '100%',
    height: 450,
    saveReport: function (args: SaveReportArgs): void {
        let reports: SaveReportArgs[] = [];
        let isSaved: boolean = false;
        if (localStorage.pivotviewReports && localStorage.pivotviewReports
            !== "") {
            reports = JSON.parse(localStorage.pivotviewReports);
        }
        if (args.report && args.reportName && args.reportName !== '' &&
            args.reportName !== 'Sample Report') {
            reports.map(function (item: any): any {
                if (args.reportName === item.reportName) {
                    item.report = args.report; isSaved = true;
                }
            });
            if (!isSaved) {

```

```

        reports.push(args);
    }
    localStorage.pivotviewReports = JSON.stringify(reports);
},
fetchReport: function (args: FetchReportArgs): void {
    let reportCollection: string[] = [];
    let reeportList: string[] = [];
    if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "" ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
    }
    reportCollection.map(function (item: any): void {
reeportList.push(item.reportName); });
    args.reportName = reeportList;
},
loadReport: loadReport,
removeReport: function (args: RemoveReportArgs): void {
    let reportCollection: any[] = [];
    if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "" ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
    }
    for (let i: number = 0; i < reportCollection.length; i++) {
        if (reportCollection[i].reportName === args.reportName) {
            reportCollection.splice(i, 1);
        }
    }
    if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "" ) {
        localStorage.pivotviewReports =
JSON.stringify(reportCollection);
    }
},
renameReport: function (args: RenameReportArgs): void {
    let reportCollection: any[] = [];
    if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "" ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
    }
    if (args.isReportExists) {
        for (let i: number = 0; i < reportCollection.length; i++) {
            if (reportCollection[i].reportName === args.rename) {
                reportCollection.splice(i, 1);
            }
        }
    }
    reportCollection.map(function (item: any): any {
        if (args.reportName === item.reportName) {
            item.reportName = args.rename;
        }
    });
    if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "" ) {
        localStorage.pivotviewReports =
JSON.stringify(reportCollection);
    }
}

```



```

    },
    dataBound: function () {
        // Set the default report name to load it in the pivot table
        // during initial rendering.
        if (pivotObj && isInitial) {
            isInitial = false;
            pivotObj.toolbarModule.action = 'Load';
            let reportList = getInstance(select('#' + pivotObj.element.id
+ '_reportlist', pivotObj.element), DropDownList);
            (reportList as DropDownList).value = 'Default report';
            loadReport({ reportName: 'Default report' });
        }
    },
    load: function (args: any): void {
        // Save the desired report that needs to be loaded at initial
        // rendering here.
        let dataSourceSettings = {
            dataSource: pivotData as IDataset[],
            columns: [{ name: 'Year' }],
            enableSorting: true,
            allowLabelFilter: true,
            values: [{ name: 'Sold', caption: 'Units Sold' }],
            allowValueFilter: true,
            formatSettings: [{ name: 'Sold', format: 'C0' }],
            rows: [{ name: 'Country' }],
        };
        let displayOption = { view: 'Both' };
        let gridSettings = { columnWidth: 100 };
        let report = { dataSourceSettings: dataSourceSettings,
displayOption: displayOption, gridSettings: gridSettings };
        let reports = [
            {
                report: JSON.stringify(report),
                reportName: 'Default report',
            },
        ];
        localStorage['pivotviewReports'] = JSON.stringify(reports);
    },
    newReport: function (): void {
        pivotObj.setProperties({
            dataSourceSettings: { columns: [], rows: [], values: [],
filters: [] }
        }, false);
    },
    toolbarRender: function (args: ToolbarArgs): void {
        args.customToolbar.splice(6, 0, {
            type: 'Separator'
        });
        args.customToolbar.splice(9, 0, {
            type: 'Separator'
        });
    },
    toolbar: ['New', 'Save', 'SaveAs', 'Rename', 'Remove', 'Load',
'Grid', 'Chart', 'Export', 'SubTotal', 'GrandTotal', 'Formatting',
'FieldList'],
    allowExcelExport: true,
    allowConditionalFormatting: true,

```

```

        allowNumberFormatting: true,
        allowPdfExport: true,
        showToolbar: true,
        allowCalculatedField: true,
        displayOption: { view: 'Both' },
        showFieldList: true
    });
    pivotObj.appendTo('#PivotTable');
    function loadReport(args: LoadReportArgs): void {
        let reportCollection: string[] = [];
        if (localStorage.pivotviewReports && localStorage.pivotviewReports !==
        "") {
            reportCollection = JSON.parse(localStorage.pivotviewReports);
        }
        reportCollection.map(function (item: any): void {
            if (args.reportName === item.reportName) {
                args.report = item.report;
            }
        });
        if (args.report) {
            pivotObj.dataSourceSettings =
            JSON.parse(args.report).dataSourceSettings;
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Pivot Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Pivot Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  charts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  pivotview/styles/material.css" rel="stylesheet">

```

```

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <div id="PivotTable"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD009 -->

Display string value to pivot table values in EJ2 JavaScript Pivotview control

End user can display string value to the pivot table's value cell by using the [aggregateCellInfo](#) event.

In the following example, each cell value of the **Sold** field's actual value has been assigned from its combination data sets obtained from the [args.cellSets](#) in the [aggregateCellInfo](#) event.

INDEX.TS

```

import { PivotView, FieldList, IDataset, AggregateEventArgs } from
'@syncfusion/ej2-pivotview';
import { pivotData } from './datasource.ts';
PivotView.Inject(FieldList);
let pivotGridObj: PivotView = new PivotView({
    dataSourceSettings: {
        expandAll: false,
        dataSource: pivotData as IDataset[],
        columns: [{ name: 'Year' }, { name: 'Quarter' }],
        values: [{ name: 'Sold' }, { name: 'Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }]
    },
    showFieldList: true,
    width: '100%',
    height: 500,
    aggregateCellInfo: (args: AggregateEventArgs) => {
        if (args.fieldName === 'Sold') {
            args.value = secondsToHms(args.value);
        }
    }
}

```

```

});
pivotGridObj.appendTo('#PivotView');
function secondsToHms(d: number) {
    d = Number(d);
    var h = Math.floor(d / 3600);
    var m = Math.floor((d % 3600) / 60);
    var s = Math.floor((d % 3600) % 60);
    return (
        ('0' + h).slice(-2) + ':' + ('0' + m).slice(-2) + ':' + ('0' +
s).slice(-2)
    );
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Pivot Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Pivot Grid Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
pivotview/styles/material.css" rel="stylesheet">

    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div>
    <div id="PivotTable"></div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Predefined Dialogs

Draggable in EJ2 JavaScript Predefined dialogs control

The predefined dialogs supports dragging within its target container by grabbing the dialog header, which allows the user to reposition the dialog dynamically by using `isDraggable` property.

Alert drag

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let alertBtn: Button = new Button({cssClass: 'e-danger'});
alertBtn.appendTo('#alertBtn');
document.getElementById('alertBtn').onclick = (): void => {
  // Initialize and render alert dialog
  DialogUtility.alert({
    title: 'Low Battery',
    width: '250px',
    isDraggable: true,
    content: '10% of battery remaining',
  })
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="alertBtn" type="button">Alert</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Confirm drag

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let confirmBtn: Button = new Button({cssClass: 'e-success'});
confirmBtn.appendTo('#confirmBtn');
document.getElementById('confirmBtn').onclick = (): void => {
    DialogUtility.confirm({
        title: 'Delete Multiple Items',
        content: "Are you sure you want to permanently delete these items?",
        width: '300px',
        isDraggable: true
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="confirmBtn" type="button">Confirm</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prompt drag

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let promptBtn: Button = new Button({isPrimary: true});
promptBtn.appendTo('#promptBtn');
document.getElementById('promptBtn').onclick = (): void => {
    DialogUtility.confirm({
        title: 'Join Chat Group',
        width: '250px',
        isDraggable: true,
        content: '<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />',
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">

```

```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="promptBtn" type="button">Prompt</button>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Animation in EJ2 JavaScript Predefined dialogs control

The predefined dialogs can be animated during the open and close actions. Also, user can customize animation's **delay**, **duration** and **effect** of animation by using the **animationSettings** property.

In the below sample, **Zoom** effect is enabled. So, The Dialog will open with **ZoomIn** and close with **ZoomOut** effects.

Alert animation

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let alertBtn: Button = new Button({cssClass: 'e-danger'});
alertBtn.appendTo('#alertBtn');
document.getElementById('alertBtn').onclick = (): void => {
  // Initialize and render alert dialog
  DialogUtility.alert({
    title: 'Low Battery',
    width: '250px',
    animationSettings: { effect: 'Zoom' },
    content: '10% of battery remaining'
  })
};

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="alertBtn" type="button">Alert</button>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Confirm animation

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let confirmBtn: Button = new Button({cssClass: 'e-success'});
confirmBtn.appendTo('#confirmBtn');
document.getElementById('confirmBtn').onclick = (): void => {
  DialogUtility.confirm({
    title: 'Delete Multiple Items',
    content: "Are you sure you want to permanently delete these items?",
    width: '300px',
    animationSettings: { effect: 'Zoom' }
  });
};

```

```
});
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="confirmBtn" type="button">Confirm</button>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Prompt animation

INDEX.TS

```
import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let promptBtn: Button = new Button({isPrimary: true});
promptBtn.appendTo('#promptBtn');
document.getElementById('promptBtn').onclick = (): void => {
  DialogUtility.confirm({
    title: 'Join Chat Group',
    width: '250px',
```

```

        animationSettings: { effect: 'Zoom' }
        content: '<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />',
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="promptBtn" type="button">Prompt</button>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Position in EJ2 JavaScript Predefined dialogs control

Customize the dialog position by using the `position` property. The position can be represented with specific `X` and `Y` values.

- The `PositionDataModel.X` can be configured with a left, center, right, or offset value. By default, the value is set as center.

- The `PositionDataModel.Y` can be configured with a top, center, bottom, or offset value. By default, the value is set as `center`.

Alert position

INDEX.TS

```
import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let alertBtn: Button = new Button({cssClass: 'e-danger'});
alertBtn.appendTo('#alertBtn');
document.getElementById('alertBtn').onclick = (): void => {
    // Initialize and render alert dialog
    DialogUtility.alert({
        title: 'Low Battery',
        width: '250px',
        content: '10% of battery remaining',
        position: { X: 'center', Y: 'center' }
    })
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="alertBtn" type="button">Alert</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Confirm position

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let confirmBtn: Button = new Button({cssClass: 'e-success'});
confirmBtn.appendTo('#confirmBtn');
document.getElementById('confirmBtn').onclick = (): void => {
  DialogUtility.confirm({
    title: 'Delete Multiple Items',
    content: "Are you sure you want to permanently delete these items?",
    width: '300px',
    position: { X: 'center', Y: 'center' }
  });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="confirmBtn" type="button">Confirm</button>
  </div>
</body>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prompt position

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let promptBtn: Button = new Button({isPrimary: true});
promptBtn.appendTo('#promptBtn');
document.getElementById('promptBtn').onclick = (): void => {
    DialogUtility.confirm({
        title: 'Join Chat Group',
        width: '250px',
        content: '<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />',
        position: { X: 'center', Y: 'center' }
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <button id="promptBtn" type="button">Prompt</button>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dimension in EJ2 JavaScript Predefined dialogs control

Customize the predefined dialogs dimensions using the **height** and **width** properties. You can specify the dimension values in both pixels and percentage format to change the default dialog width and height values.

Alert dimension

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let alertBtn: Button = new Button({cssClass: 'e-danger'});
alertBtn.appendTo('#alertBtn');
document.getElementById('alertBtn').onclick = (): void => {
    // Initialize and render alert dialog
    DialogUtility.alert({
        title: 'Not enough space',
        width: '300px',
        height: '200px',
        content: 'Delete certain files to free up space to store more
items.',
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="alertBtn" type="button">Alert</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Confirm dimension

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let confirmBtn: Button = new Button({cssClass: 'e-success'});
confirmBtn.appendTo('#confirmBtn');
document.getElementById('confirmBtn').onclick = (): void => {
    DialogUtility.confirm({
        title: 'Delete Multiple Items',
        content: "Are you sure you want to permanently delete these items?",
        width: '300px',
        height: '200px'
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="confirmBtn" type="button">Confirm</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prompt dimension

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let promptBtn: Button = new Button({isPrimary: true});
promptBtn.appendTo('#promptBtn');
document.getElementById('promptBtn').onclick = (): void => {
    DialogUtility.confirm({
        title: 'Join Chat Group',
        width: '300px',
        height: '250px',
        content: '<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />',
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="promptBtn" type="button">Prompt</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Max-width and max-height

To have a restricted max-width and max-height dialog dimension, you need to specify the max-width, max-height CSS properties for the component's container element by using the `cssClass` property. The max-height value is calculated in source level and set to the dialog. so, need to override the max-height property.

Use the following code to customize the max-width and max-height for alert dialog:

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let alertBtn: Button = new Button({cssClass: 'e-danger'});
alertBtn.appendTo('#alertBtn');
document.getElementById('alertBtn').onclick = (): void => {
    // Initialize and render alert dialog
    DialogUtility.alert({
        title: 'About SYNCFUSION Succinctly Series',
        content: 'In the Succinctly series, Syncfusion created a robust,
        free library of more than 130 technical e-books formatted for PDF, Kindle,
        and EPUB.Each title in the Succinctly series is written by a carefully
        chosen expert and provides essential content in about 100 pages. The
        Succinctly series was born in 2012 out of a desire to provide concise
        technical e-books for software developers.',
        cssClass : 'customClass'
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="alertBtn" type="button">Alert</button>
  </div>
  <style>
    .e-dialog.customClass {
      max-width: 350px;
      max-height: 250px !important;
    }

  </style>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Min-width and min-height

To have a restricted min-width and min-height dialog dimension, you need to specify the min-width, min-height CSS properties for the component's container element by using the `cssClass` property.

Use the following code to customize the min-width and min-height for alert dialog:

INDEX.TS

```
import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let alertBtn: Button = new Button({cssClass: 'e-danger'});
alertBtn.appendTo('#alertBtn');
document.getElementById('alertBtn').onclick = (): void => {
    // Initialize and render alert dialog
    DialogUtility.alert({
        title: 'About SYNCFUSION Succinctly Series',
        content: 'The Succinctly series was born in 2012.',
        cssClass: 'customClass'
    });
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="alertBtn" type="button">Alert</button>
  </div>
  <style>
    .e-dialog.customClass {
      min-width: 300px;
      min-height: 200px;
    }
  </style>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization in EJ2 JavaScript Predefined dialogs control

You can customize the predefined dialogs buttons by using below properties.

- **okButton** - Use this property to customize **OK** button text.
- **cancelButton** - Use this property to customize **Cancel** button text.

Use the following code snippet for **alert**, **confirm** and **prompt** to customize the predefined dialogs action buttons.

For alert dialog , customized the default dialog button content as **Dismiss** by using the **text** property.

For confirm dialog, customized the default dialog buttons content as **Yes** and **No** by using the **text** property and also customized the dialog button icons by using **icon** property.

For prompt dialog , customized the default dialog buttons content as **Connect** and **Close** by using **text** property.

Alert action button

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let alertBtn: Button = new Button({cssClass: 'e-danger'});
alertBtn.appendTo('#alertBtn');
document.getElementById('alertBtn').onclick = (): void => {
    // Initialize and render alert dialog
    DialogUtility.alert({
        title: 'Good job!',
        content: 'You clicked the alert button!',
        width: '250px',
        okButton: { text: 'Dismiss' }
    })
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="alertBtn" type="button">Alert</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Confirm action buttons

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let confirmBtn: Button = new Button({cssClass: 'e-success'});
confirmBtn.appendTo('#confirmBtn');
document.getElementById('confirmBtn').onclick = (): void => {
    DialogUtility.confirm({
        title: 'Delete File',
        content: "Are you sure you want to permanently delete these file?",
        width: '300px',
        okButton: { text: 'Yes', icon: 'e-icons e-check' },
        cancelButton: { text: 'No', icon: 'e-icons e-close' }
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="confirmBtn" type="button">Confirm</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prompt action buttons

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let promptBtn: Button = new Button({isPrimary: true});
promptBtn.appendTo('#promptBtn');
document.getElementById('promptBtn').onclick = (): void => {
    DialogUtility.confirm({
        title: 'Join Chat Group',
        width: '300px',
        content: '<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />',
        okButton: { text: 'Connect' },
        cancelButton: {text: 'Close' }
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">

```

```

    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="promptBtn" type="button">Prompt</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show or hide dialog close button

When rendering the predefined dialogs through utility methods, You can close the dialog using the following ways. The default values of `closeOnEscape` and `showCloseIcon` is `false`.

- By pressing the escape key if the [closeOnEscape](#) property is enabled.
- By clicking the close button if the [showCloseIcon](#) property is enabled.

You can also manually close the Dialogs by creating an instance to the dialog and call the [hide](#) method.

Use the following code for **alert**, **confirm** and **prompt** to demonstrates the different ways of hiding the utility dialog.

Alert dialog close button

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let alertBtn: Button = new Button({cssClass: 'e-danger'});
alertBtn.appendTo('#alertBtn');
document.getElementById('alertBtn').onclick = (): void => {
    // Initialize and render alert dialog
    DialogUtility.alert({

```



```

        title: 'Low Battery',
        content: '10% of battery remaining',
        width: '250px',
        showCloseIcon : true,
        closeOnEscape : true
    })
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Dialog utility</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript UI Components">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="alertBtn" type="button">Alert</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Confirm dialog close button

INDEX.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let confirmBtn: Button = new Button({cssClass: 'e-success'});

```

```
confirmBtn.appendTo('#confirmBtn');
document.getElementById('confirmBtn').onclick = (): void => {
    DialogUtility.confirm({
        title: 'Delete File',
        content: "Are you sure you want to permanently delete these items?",
        width: '300px',
        showCloseIcon : true,
        closeOnEscape : true
    });
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="confirmBtn" type="button">Confirm</button>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

Prompt dialog close button

INDEX.TS

```
import { DialogUtility } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let promptBtn: Button = new Button({isPrimary: true});
promptBtn.appendTo('#promptBtn');
document.getElementById('promptBtn').onclick = (): void => {
    DialogUtility.confirm({
        title: 'Join Chat Group',
        width: '300px',
        content: '<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />',
        showCloseIcon : true,
        closeOnEscape : true
    });
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="promptBtn" type="button">Prompt</button>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customize dialog content

You can load custom content in predefined dialogs using the `content` property.

Use the following code to customize the dialog content to render the custom TextBox component inside the prompt dialog to get the username from the user.

INDEX.TS

```
import { DialogUtility } from '@syncfusion/ej2-popups';
import { TextBox } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
let promptBtn: Button = new Button({isPrimary: true});
promptBtn.appendTo('#promptBtn');
document.getElementById('promptBtn').onclick = (): void => {
    dialogObj = DialogUtility.confirm({
        title: 'Join Chat Group',
        content: '<p>Enter your name: </p><input id="name" />',
        width: '350px'
    });
    let inputobj1: TextBox = new TextBox({
        placeholder: 'Type here..'
    });
    inputobj1.appendTo('#name');
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Dialog utility</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript UI Components">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="promptBtn" type="button">Prompt</button>
```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Progressbar

Types in EJ2 JavaScript Progressbar control

Visualize progress in different shapes (rectangle, circle, and semi-circle) to give a unique appearance to your app design.

Linear

Set **type** to Linear to get the linear progress bar. It also support secondary progress and different mode of progress.

INDEX.TS

```

import { ProgressBar } from '@syncfusion/ej2-progressbar';
let uploadProgress: ProgressBar = new ProgressBar({
    type: 'Linear',
    height: '60',
    value: 100,
    animation: {
        enable: true,
        duration: 2000,
        delay: 0,
    }
});
uploadProgress.appendTo('#lineardeterminate');
let successProgress: ProgressBar = new ProgressBar({
    type: 'Linear',
    height: '60',
    isIndeterminate: true,
    value: 20,
    animation: {
        enable: true,
        duration: 2000,
        delay: 0,
    },
});
successProgress.appendTo('#linearindeterminate');
let warningsProgress: ProgressBar = new ProgressBar({
    type: 'Linear',
    height: '60',
    value: 40,
    secondaryProgress: 60,
    animation: {
        enable: true,
        duration: 2000,
        delay: 0,
    }
});

```

```

    },
  });
  warningsProgress.appendTo('#linearbuffer');
  let errorProgress: ProgressBar = new ProgressBar({
    type: 'Linear',
    height: '60',
    segmentCount: 8,
    value: 100,
    animation: {
      enable: true,
      duration: 2000,
      delay: 0,
    },
  });
  errorProgress.appendTo('#linearsegment');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="lineardeterminate" class="linear-progress"></div>
      <div id="linearindeterminate" class="linear-progress"></div>
      <div id="linearbuffer" class="linear-progress"></div>
      <div id="linearsegment" class="linear-progress"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Circular

Set **type** to Circular to get the circular progress bar. It also support secondary progress and different mode of progress.

INDEX.TS

```
import { ProgressBar } from '@syncfusion/ej2-progressbar';
let uploadProgress: ProgressBar = new ProgressBar({
  type: 'Circular',
  height: '60',
  value: 100,
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
  }
});
uploadProgress.appendTo('#lineardeterminate');
let successProgress: ProgressBar = new ProgressBar({
  type: 'Circular',
  height: '60',
  isIndeterminate: true,
  value: 20,
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
  },
});
successProgress.appendTo('#linearindeterminate');
let warningsProgress: ProgressBar = new ProgressBar({
  type: 'Circular',
  height: '60',
  value: 40,
  secondaryProgress: 60,
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
  },
});
warningsProgress.appendTo('#linearbuffer');
let errorProgress: ProgressBar = new ProgressBar({
  type: 'Circular',
  height: '60',
  segmentCount: 8,
  value: 100,
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
  },
});
errorProgress.appendTo('#linearsegment');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="lineardeterminate" class="linear-progress"></div>
      <div id="linearindeterminate" class="linear-progress"></div>
      <div id="linearbuffer" class="linear-progress"></div>
      <div id="linearsegment" class="linear-progress"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Tool tip in EJ2 JavaScript Progressbar control

The tooltip for the progress bar is used to represent the progress value. During the initial load, it can be enabled by using the [enable](#) property. The [showTooltipOnHover](#) property can show the tooltip on mouseover.

INDEX.TS

```

import { ProgressBar } from '@syncfusion/ej2-progressbar';
let uploadProgress: ProgressBar = new ProgressBar({
  type: 'Linear',
  height: '60',
  value: 100,
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
  },
  tooltip: {
    enable: true,
    showTooltipOnHover: true
  }
});

```



```
});
uploadProgress.appendTo('#lineardeterminate');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="lineardeterminate" class="linear-progress"></div>
      <div id="linearindeterminate" class="linear-progress"></div>
      <div id="linearbuffer" class="linear-progress"></div>
      <div id="linearsegment" class="linear-progress"></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Format

By default, the tooltip shows information about progress. In addition to that, show more information in the tooltip using the [format](#) property.

INDEX.TS

```
import { ProgressBar } from '@syncfusion/ej2-progressbar';
let uploadProgress: ProgressBar = new ProgressBar({
  type: 'Linear',
  height: '60',
  value: 100,
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
```

```

    },
    tooltip:{
        enable: true,
        format: "Progress: ${value}"
    }
});
uploadProgress.appendTo('#lineardeterminate');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="lineardeterminate" class="linear-progress"></div>
      <div id="linearindeterminate" class="linear-progress"></div>
      <div id="linearbuffer" class="linear-progress"></div>
      <div id="linearsegment" class="linear-progress"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

The [fill](#) and [border](#) properties are used to customize the background color and border of the tooltip respectively. The [textStyle](#) property in the tooltip is used to customize the font of the tooltip text.

INDEX.TS

```

import { ProgressBar } from '@syncfusion/ej2-progressbar';
let uploadProgress: ProgressBar = new ProgressBar({
  type: 'Linear',
  height: '60',

```

```

        value: 100,
        animation: {
            enable: true,
            duration: 2000,
            delay: 0,
        },
        tooltip: {
            enable: true,
            format: "Progress: ${value}",
            textStyle: {
                fontWeight: '600',
                size: '9px',
                color: 'red',
                fontFamily: 'Roboto',
                fontStyle: 'Italic'
            }
        }
    });
    uploadProgress.appendTo('#lineardeterminate');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="lineardeterminate" class="linear-progress"></div>
      <div id="linearindeterminate" class="linear-progress"></div>
      <div id="linearbuffer" class="linear-progress"></div>
      <div id="linearsegment" class="linear-progress"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

States in EJ2 JavaScript Progressbar control

Visualize progress in different states.

Determinate

<!-- markdownlint-disable MD033 -->

This is the default state. You can use it when the progress estimation is known.

INDEX.TS

```
import { ProgressBar } from '@syncfusion/ej2-progressbar';
let percentageProgress: ProgressBar = new ProgressBar({
    type: 'Linear',
    height: '60',
    value: 100,
});
percentageProgress.appendTo('#percentage');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="percentage"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Indeterminate

By enabling the **IsIndeterminate** property, the state of the progress bar can be changed to indeterminate when the progress cannot be estimated or is not being calculated. It can be combined with determinate mode to know that the application is estimating progress before the actual progress starts.

INDEX.TS

```
import { ProgressBar } from '@syncfusion/ej2-progressbar';
let percentageProgress: ProgressBar = new ProgressBar({
  type: 'Linear',
  height: '60',
  isIndeterminate: true,
  value: 20,
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
  },
});
percentageProgress.appendTo('#percentage');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="percentage"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Buffer

<!-- markdownlint-disable MD033 -->

You can use a secondary progress indicator when the primary progress depends on the secondary progress. This will allow users to visualize both primary and secondary progress simultaneously.

INDEX.TS

```
import { Progressbar } from '@syncfusion/ej2-progressbar';
let percentageProgress: Progressbar = new Progressbar({
  type: 'Linear',
  height: '60',
  value: 40,
  secondaryProgress: 60,
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
  },
});
percentageProgress.appendTo('#percentage');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="percentage"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization in EJ2 JavaScript Progressbar control

Segments

<!-- markdownlint-disable MD033 -->

We can divide a progress bar into multiple segments using a `segmentCount` to visualize the progress of multiple sequential tasks.

INDEX.TS

```
import { ProgressBar } from '@syncfusion/ej2-progressbar';
let errorProgress: ProgressBar = new ProgressBar({
  type: 'Circular',
  height: '60',
  segmentCount: 8,
  value: 100,
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
  },
});
errorProgress.appendTo('#percentage');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="percentage"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Thickness

Customize the thickness of the track using [trackThickness](#), progress using [progressThickness](#) and secondary progress using [secondaryProgressThickness](#) to render the progress bar with different appearances.

INDEX.TS

```
import { ProgressBar } from '@syncfusion/ej2-progressbar';
let percentageProgress: ProgressBar = new ProgressBar({
  type: 'Linear',
  height: '60',
  width: '90%',
  trackThickness: 24,
  progressThickness: 24,
  secondaryProgressThickness: 20,
  value: 100,
  showProgressValue: true,
  labelStyle: {
    color: '#FFFFFF'
  },
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
  }
});
percentageProgress.appendTo('#percentage');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="percentage"></div>
    </div>
  </div>
</script>
```



```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Radius

<!-- markdownlint-disable MD033 -->

The radius of the progress bar can be customized using **radius** property and corner can be customized by **cornerRadius** property.

INDEX.TS

```

import { ProgressBar } from '@syncfusion/ej2-progressbar';
let percentageProgress: ProgressBar = new ProgressBar({
    type: "Circular",
    value: 60,
    width: "160px",
    height: "160px",
    enableRtl: false,
    trackColor: "#FFD939",
    radius: "100%",
    progressColor: "white",
    trackThickness: 80,
    cornerRadius: "Round",
    progressThickness: 10,
    animation: {
        enable: true,
        duration: 2000,
        delay: 0
    }
});
percentageProgress.appendTo("#percentage");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Toast</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Toolbar Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div class="row linear-parent">
        <div id="percentage"></div>
      </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

InnerRadius

<!-- markdownlint-disable MD033 -->

The inner radius of the progress bar can be customized using `innerRadius` property.

INDEX.TS

```

import { ProgressBar } from '@syncfusion/ej2-progressbar';
let percentageProgress: ProgressBar = new ProgressBar({
  type: 'Circular',
  value: 60,
  width: '160px',
  height: '160px',
  enableRtl: false,
  trackColor: '#FFD939',
  radius: '100%',
  innerRadius: '80%',
  progressColor: 'white',
  trackThickness: 80,
  cornerRadius: 'Round',
  progressThickness: 10,
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
  },
});
percentageProgress.appendTo('#percentage');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="row linear-parent">
            <div id="percentage"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Progress color and track color

<!-- markdownlint-disable MD033 -->

Customize the color of progress, secondary progress, and track by using the [progressColor](#), [secondaryProgressColor](#), and [trackColor](#) properties.

INDEX.TS

```

import { Progressbar } from '@syncfusion/ej2-progressbar';
let percentageProgress: Progressbar = new Progressbar({
    type: 'Linear',
    height: '60',
    width: '90%',
    trackThickness: 24,
    progressThickness: 24,
    cornerRadius: 'Round',
    progressColor: '#E3165B',
    trackColor: '#F8C7D8',
    secondaryProgressColor: 'green',
    secondaryProgress: 70,
    value: 50,
    showProgressValue: true,
    labelStyle: {
        color: 'FFFFFF'
    },
    animation: {
        enable: true,
        duration: 2000,
        delay: 0,
    },
    textRender: (args: ITextRenderEventArgs) => {
        args.text = '50';
    }
});

```

```
percentageProgress.appendTo('#percentage');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="percentage"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Annotation in EJ2 JavaScript Progressbar control

Annotation

In the circular progress bar, you can add any view to the center using the **Content** property in annotation.

For example, you can include add, start, or pause button to control the progress. You can also add an image that indicates the actual task in progress or add custom text that conveys how far the task is completed.

INDEX.TS

```
import { ProgressBar, ProgressAnnotation } from "@syncfusion/ej2-
progressbar";
ProgressBar.Inject(ProgressAnnotation);
let percentageProgress: ProgressBar = new ProgressBar({
  type: "Circular",
  innerRadius: "190%",
  trackThickness: 80,
  cornerRadius: "Round",
```

```

trackColor: "#FFD939",
annotations: [
  {
    content:
      '<div id="point1" style="font-size:20px;font-weight:bold;color:#ffffff;fill:#ffffff"><span>60%</span></div>'
  }
]
});
percentageProgress.appendTo("#percentage");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="percentage"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Label

You can show the progress value in both linear and circular progress bar using **showProgressValue** property.

INDEX.TS

```

import { ProgressBar, ITextRenderEventArgs } from "@syncfusion/ej2-
progressbar";
let percentageProgress: ProgressBar = new ProgressBar({
  type: 'Linear',

```

```

        trackThickness: 24,
        progressThickness: 24,
        value: 50,
        showProgressValue: true,
        labelStyle: {
            color: '#FFFFFF'
        },
        animation: {
            enable: true,
            duration: 2000,
            delay: 0,
        },
        textRender: (args: ITextRenderEventArgs) => {
            args.text = '50';
        }
    });
    percentageProgress.appendTo('#percentage');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="percentage"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Animation in EJ2 JavaScript Progressbar control

<!-- markdownlint-disable MD033 -->

Progress Bar support to animate the progress by using **animation** property. Enable the animation by setting **enable** property and also you can control the speed by using **duration** property.

INDEX.TS

```
import { ProgressBar } from '@syncfusion/ej2-progressbar';
let percentageProgress: ProgressBar = new ProgressBar({
  type: 'Linear',
  trackThickness: 24,
  progressThickness: 24,
  value: 100,
  showProgressValue: true,
  animation: {
    enable: true,
    duration: 2000,
    delay: 0
  }
});
percentageProgress.appendTo('#percentage');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="percentage"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Range in EJ2 JavaScript Progressbar control

<!-- markdownlint-disable MD033 -->

Range represents the entire span of the ProgressBar and can be defined using the **minimum** and **maximum** properties.

INDEX.TS

```
import { ProgressBar } from '@syncfusion/ej2-progressbar';
let uploadProgress: ProgressBar = new ProgressBar({
  type: 'Linear',
  trackThickness: 24,
  progressThickness: 24,
  value: 90,
  minimum: 10,
  maximum: 90,
  showProgressValue: true,
  cornerRadius: 'Round',
  labelStyle: {
    color: '#FFFFFF'
  },
  animation: {
    enable: true,
    duration: 2000,
    delay: 0,
  }
});
uploadProgress.appendTo('#percentage');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toast</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="percentage"></div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
```



```
if(ele) {  
    ele.style.visibility = "visible";  
}  
  
</script>  
<script src="index.js" type="text/javascript"></script>  
</body></html>
```

Theme in EJ2 JavaScript Progressbar control

To enhance the visualization of the Progress Bar, use different themes that are mapped to the [theme](#) property. The following themes are available in the Progress Bar.

- Material
- Fabric
- Bootstrap
- Bootstrap 4
- High Contrast
- Tailwind

INDEX.TS

```
import { ProgressBar } from '@syncfusion/ej2-progressbar';  
let linearThemeProgress: ProgressBar = new ProgressBar({  
    type: 'Linear',  
    height: '60',  
    width: '100%',  
    value: 40,  
    secondaryProgress: 60,  
    trackThickness: 20,  
    progressThickness: 20,  
    theme: 'HighContrast',  
    animation: {  
        enable: true,  
        duration: 2000,  
        delay: 0  
    }  
});  
linearThemeProgress.appendTo('#linearTheme');  
let circularThemeProgress: ProgressBar = new ProgressBar({  
    type: 'Circular',  
    height: '150',  
    width: '100%',  
    value: 40,  
    secondaryProgress: 60,  
    trackThickness: 20,  
    progressThickness: 20,  
    theme: 'HighContrast',  
    animation: {  
        enable: true,  
        duration: 2000,  
        delay: 0  
    }  
});  
circularThemeProgress.appendTo('#circularTheme');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Progress Bar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="row linear-parent">
      <div id="linearTheme" class="linear-progress"></div>
      <div id="circularTheme" class="linear-progress"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Events in EJ2 JavaScript Progressbar control

This section describes the Progress Bar events that will be triggered when appropriate actions are performed. Following events are available in the Progress Bar.

- [valueChanged](#) - When the progress value changes, this event is triggered.
- [progressCompleted](#) - When the progress reaches the maximum value, this event is triggered.
- [animationComplete](#) - When animation is enabled and the progress to the value is reached, this event is triggered.
- [annotationRender](#) - Before each annotation is rendered, this event is triggered.
- [textRender](#) - Before the progress text is rendered, this event is triggered
- [load](#) - When the control begins to render, this event is triggered.
- [loaded](#) - After the control has been rendered, this event is triggered.
- [resized](#) - When the window is resized, this event is triggered.

INDEX.TS

```

import {ProgressBar, ProgressAnnotation, IProgressValueEventArgs,
ITextRenderEventArgs, IAnnotationRenderEventArgs,

```

```

ILoadedEventArgs, IProgressResizeEventArgs } from '@syncfusion/ej2-
progressbar';
ProgressBar.Inject(ProgressAnnotation);
let progress: ProgressBar = new ProgressBar({
  type: 'Linear',
  trackThickness: 24,
  progressThickness: 24,
  value: 50,
  labelStyle: {
    color: '#FFFFFF'
  },
  animation: {
    enable: true,
    duration: 2000,
    delay: 0
  },
  valueChanged: (args: IProgressValueEventArgs) => {
    // Here you can customize the code.
  },
  progressCompleted: (args: IProgressValueEventArgs) => {
    // Here you can customize the code.
  },
  animationComplete: (args: IProgressValueEventArgs) => {
    // Here you can customize the code.
  },
  textRender: (args: ITextRenderEventArgs) => {
    // Here you can customize the code.
  },
  annotationRender: (args: IAnnotationRenderEventArgs) => {
    // Here you can customize the code.
  },
  load: (args: ILoadedEventArgs) => {
    // Here you can customize the code.
  },
  loaded: (args: ILoadedEventArgs) => {
    // Here you can customize the code.
  },
  resized: (args: IProgressResizeEventArgs) => {
    // Here you can customize the code.
  }
});
progress.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Progress Bar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="row linear-parent">
            <div id="element"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in EJ2 JavaScript Progress bar control

The Progress bar control followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Progress bar control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

```

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

```

WAI-ARIA attributes

The Progress bar control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Progress bar control:

- progressbar (role)
- aria-valuemin (attribute)
- aria-valuemax (attribute)
- aria-valuenow (attribute)
- aria-label (attribute)

Keyboard interaction

The Progress bar control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Progress bar control.

| **Press** | **To do this** |

| --- | --- |

| **Tab** | Moves the focus to the Progress bar element. |

| **Ctrl + P** | Prints the Progress bar. |

Ensuring accessibility

The Progress bar control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Progress bar control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Progress bar control with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript controls](#)

ProgressBar

<!-- markdownlint-disable MD002 MD022 -->

Spinner and progress in EJ2 JavaScript Progress button control

Change spinner position

Spinner position can be changed by modifying the [position](#) property of [spinSettingsModel](#). By default, the spinner is positioned at the left of the ProgressBar. You can position it at the **left**, **right**, **top**, **bottom**, or **center** of the text content.

Change spinner size

Spinner size can be changed by modifying the [width](#) property of [spinSettingsModel](#). In this demo, the **width** is set to **20** to change the spinner size.

Spinner template

You can use custom spinner by specifying the [template](#) property of [spinSettingsModel](#) with custom styles.

The following sample demonstrates the above functionalities of the spinner.

INDEX.TS

```
import { ProgressBar } from '@syncfusion/ej2-splitbuttons';
let progressBtn: ProgressBar = new ProgressBar({ content: 'Submit',
spinSettings: { position: 'Right', width: 20, template: '<div
class="template"></div>' } });
progressBtn.appendTo('#progressbtn');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ProgressBar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="This sample is to customize spinner
width, template and position in the progress or spin button">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

        <button id="progressbtn"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

@keyframes custom-rolling {
    0% {
        -webkit-transform: rotate(0deg);
        transform: rotate(0deg);
    }
    100% {
        -webkit-transform: rotate(360deg);
        transform: rotate(360deg);
    }
}

.template {
    border: 2px solid green;
    border-style: dotted;
    border-radius: 50%;
    border-top-color: transparent;
    border-bottom-color: transparent;
    height: 16px;
    width: 16px;
}

.template {
    -webkit-animation: custom-rolling 1.3s linear infinite;
    animation: custom-rolling 1.3s linear infinite;
}

#container {
    visibility: hidden;
}

#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Progress

Content animation

The [content](#) of the ProgressButton can be animated during progress using the [effect](#) property of [animationSettingsModel](#). You can also set custom duration and timing function using the [duration](#) and

[easing](#) properties. The possible [effect](#) values are [None](#), [SlideLeft](#), [SlideRight](#), [SlideUp](#), [SlideDown](#), [ZoomIn](#), and [ZoomOut](#).

INDEX.TS

```
import { ProgressButton } from '@syncfusion/ej2-splitbuttons';
let progressBtn: ProgressButton = new ProgressButton({ content: 'Slide
Left', enableProgress: true, animationSettings: { effect: 'SlideLeft',
duration: 500, easing: 'linear' }, spinSettings: { position: 'Center' } });
progressBtn.appendTo('#progressbtn');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ProgressButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="This demo for Essential JS2 Progress
button has progress indicator and spinner. It supports texts, icons, styles,
sizes, positions, and its customization.">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="progressbtn"></button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
@font-face {
  font-family: 'btn-icon';
  src:
```



```

url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXNnH+dzAAABoAAAAEJnbHlmlv4
8pAAAAfgAAQYAGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAAYAAAAA
gbG9jYQn6ApQAAAHkAAAAEmlheHABFQCqAAABCAAAACBuYw1l07lFxAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGaABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAQAABAAAAQAABJ1LUzF8
PPPUACwQAAAAAANG+nFMAAAAAA2D6cUwAAAAAD9AP0AAAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAABAAAAAABAAAAAQA AAA
EAAAABAAAAQA AAAEAAAAAABAAAAQA AAAAACA AAAAwAAABQA AwABAAAAFAAEAC4AAAAEAAQAQA
A5wb//wAA5wD//wAAAAEABAAAAEAAgADAAQABQAGAAcAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAALIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAA4sD9AACAAATARF0AxcCAP4MA+gAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgwBAGMFBQcICQkLCwwMDQ4NAtONDg0MDAsLCQkIBwUFawIBAQIDBQUHCAkJCws
MDA0ODf0mDQ4NDawLCwkJCACFBQMCA239Jg4NDQ0LCwsJCQgHBQUdAgEBAGMFBQcICQkLCwsNDQ0
OAtOoDQ0NCwsLCQkIBwUFawIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MdxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEVBt8LLwg3NT8INS8FNT8NNS8JByU/BDUvCyMPAQytrQH5AgoEAQEBA RG
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAgE
BAxYLBQQEAWMCAGIEBAoBAQEECgCHBgUFBAMDAQEBAQQFBwkFBQUGef6tDwKEAwIBQMDCGwVawc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFawIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXmz8KNScxBxEfDjSBHQEfDTMhMz8OES8PiZ0BLw4hA0EDBQQ
DAQIEbf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQGL7Y0EawQCAgIBAYYKChEQDQsJCACeBAU
CYt8BAQIDBAUFBQCHBgICQgKjQECAgMEBAUFBgYHBgcIBwGcCAChBwYGBgUFBAQDAgIBAQEBAGI
DBAQFBQYGBgcHBwgmAQMDAwUFBgYHBwgICQkJ/tQCiWMEbf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAGMDAwTV+94BAQECAwMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQEAWI
CUAGHBwCGBgYFBQQEAWMBAGIBAwMEBAUFBQcGBwCHCAImCAChBwYGBgUFBAQDAgIBAdUJCQgICAg
GBwYFBQDAgEBAAAAAIAAAAAA6cD9AADAawAADchNSElAQcJAScBESNZAO78sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAA AAAEAAAAAABAAAgAAQABAAAAA
CAACACQABAAAAAADAAgAEABAAAAA AAAGAGAABAAAAAFAAsAIAABAAAAAAGAAgAKwABAAA
AAAAKACwAMwABAAAAAALABIAxwADAAEECQAAAAIACQADAAEECQABABAAcWADAAEECQACAA4AgwA
DAAEECQADABAAKQADAAEECQAEABAAoQADAAEECQAFABYasQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmlpY29uVmVyc2lvbiAxLjB
idG4taWNvbkbZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAyYgB0AG4ALQBpAGM
AbwBuAGIAdABuAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMABwB
uAEYABwBuAHQAIAABnAGUAbgBlAHIAyYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAEQBuAGMAZgBlAHM
AaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAEQBuAGMAZgBlAHMAaQB
vAG4ALgBjAG8AbQAAAAACAAAAA AAAA oAAAAA AAAAAAAAAAAAAAAAAAAAAAAGBAGEDAQQBBQE
GAQCBCAEJAAPtZWRpYS1wbGF5C21lZGlhLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrc2S0
tLTaxBGNvcHkQLWRvd25sb2FkLTayLXdmlQAQ) format('trueType');

font-weight: normal;
font-style: normal;
}
.e-btn-sb-icon {
font-family: 'btn-icon' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-download::before {

```

```

    content: '\e706';
  }
  .e-play::before {
    content: '\e700';
  }
  .e-pause::before {
    content: '\e701';
  }
}
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}

```

Change step of the ProgressButton

The progress can be visualized at the specified interval by changing the [step](#) property in the [begin](#) event of the ProgressButton. In this demo, the [step](#) property is set to [20](#) to show progress at every 20% increment.

INDEX.TS

```

import { ProgressButton, ProgressEventArgs } from '@syncfusion/ej2-splitbuttons';
let progressBtn: ProgressButton = new ProgressButton({
  content: 'Progress Step', enableProgress: true,
  begin: (args: ProgressEventArgs) => {
    args.step = 20;
  }, cssClass: 'e-hide-spinner'
});
progressBtn.appendTo('#progressbtn');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ProgressButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="This demo for Essential JS2 Progress button has progress indicator and spinner. It supports texts, icons, styles, sizes, positions, and its customization.">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

```

```
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="progressbtn"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
@font-face {
    font-family: 'btn-icon';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXNlH+dzAAABoAAAAEJnbHlmlv4
8pAAAAfgAAQYAGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAAYAAAA
gbG9jYQYQNApQAAAHKAAAAEm1heHABFQCqAAABCAAAACBuYw1l07lFxAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGaABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAACAAABAAAAQAAJ1LUzF8
PPPUACwQAAAAAANG+nFMAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAEAAgADAAQABQAGAAcAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAAlIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAAA4sD9AACAAATARF0AxcGAP4MA+gAAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgwBAGMFBQcICQkLCwwMDQ4NAtONDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA0ODf0mDQ4NDawLCwkJCACFBQMCA239Jg4NDQ0LCwsJCQgHBQUdAgEBAgMFBQcICQkLCwsNDQ0
OAtODQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEVBt8LLwg3NT8INS8FNT8NNS8JByU/BDUvCyMPAQytrQH5AgoEAQEBArg
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAgE
BAxYLBQQEAWMCAgIEBAoBAQEECgcHBgUFBAMDAQEBAQQFBwkFBQUGEf6tDwKEAwIBAQMDCgwVAwc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAghCAcGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAGBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXmZ8KNScxBxEfDjSBHQEfDTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQGL7Y0EAWQCAgIBAYYKChEQDQsJCAcEBAU
CYt8BAQIDBAUFBQcHBwgICQgKjQECAgMEBAUFBgYHBgcIBwGcCAChBwYGBgUFBQDAGIBAQEBAgI
DBAQFBQYGBgcHBwgMAQMDAwUFBgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAGMDAwTV+94BAQECAwMDBAGyAQECBAYHCAgJCgkQCAQC6/47CQkICQcIBwYGBQQEAWI
CUAGHBwcGBgYFBQQEAWMBAgIBAwMEBAUFBQcGBwCHCAImCAChBwYGBgUFBQDAGIBAdUJCQgICAg
GBwYFBQDAGEBAAAAAIAAAAAA6cD9AADAawAADchNSElAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAABAAAAAABAAAAAABAAgAAQABAAAAA
```

```

CAACACQABAAAAAADAAgAEABAAAAAAAEAAgAGAABAAAAAAFAAsAIAABAAAAAAAGAAgAKwABAAA
AAAAKACwAMwABAAAAAALABIAxwADAAEECQAAAAIAcQADAAEECQABABAAcwADAAEECQACAA4AgwA
DAAEECQADABAAKQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bi1pY29uVmVyc2lubiAxLjB
idG4taWNvbKZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1vd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMAbwBuAFIAZQBnAHUAbABhAHIAyGyB0AG4ALQBpAGM
AbwBuAGIAAdABuAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMAbwB
uAEYAbwBuAHQAIAbnAGUAbgBlAHIAyQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgBlAHM
AaQBvAG4AIAbnAGUAdABYAG8AIAbTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgBlAHMAaQB
vAG4ALgBjAG8AbQAAAAACAAAAAAAOAAAAAAGBAGEDAQBBQE
GAQcBCAEJAaptZWRpYS1wbGF5C21lZG1hLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AjbGlrc2S0
tLTaxBGNvcHkQLWRvd25sb2FkLTAYLXdmLQAA) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-btn-sb-icon {
    font-family: 'btn-icon' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-download::before {
    content: '\e706';
}
.e-play::before {
    content: '\e700';
}
.e-pause::before {
    content: '\e701';
}
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

The class `e-hide-spinner` hides the spinner in the ProgressButton, For more information, see [hide spinner](#) section.

Change progress dynamically

The progress can be changed dynamically by modifying the `percent` property in the ProgressButton events. In this demo, on 40% completion of progress, the `percent` property is set to 90 to show dynamic change of the progress.

INDEX.TS

```
import { ProgressButton, ProgressEventArgs } from '@syncfusion/ej2-splitbuttons';
let progressBtn: ProgressButton = new ProgressButton({
  content: 'Progress', enableProgress: true, duration: 15000,
  begin: (args: ProgressEventArgs) => {
    progressBtn.content = 'Progress ' + args.percent + '%';
    progressBtn.dataBind();
  },
  progress: (args: ProgressEventArgs) => {
    progressBtn.content = 'Progress ' + args.percent + '%';
    progressBtn.dataBind();
    if (args.percent === 40) {
      args.percent = 90
    }
  },
  end: (args: ProgressEventArgs) => {
    progressBtn.content = 'Progress ' + args.percent + '%';
    progressBtn.dataBind();
  },
  cssClass: 'e-hide-spinner'
});
progressBtn.appendTo('#progressbtn');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ProgressButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="This demo for Essential JS2 Progress
button has progress indicator and spinner. It supports texts, icons, styles,
sizes, positions, and its customization.">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="progressbtn"></button>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
@font-face {
    font-family: 'btn-icon';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAkAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXDNH+dzAAABoAAAAEJnbHlm1v4
8pAAAAfgAAQYAGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAARAAAAACRobXR4IAAAAAAYAAAA
gbG9jYQYQNApQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYW1l07lFxAAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGABAAAAEAAAAAFwEAAAAAAD9AABAAAAAIAAAACAAABAAAAQAAJ1LUzF8
PPPUACwQAAAAAANg+nFMAAAAA2D6cUwAAAAAD9AP0AAACAAACAAAAAIAAAIAJ4AAwAAAA
AAAgAAAAoACgAAAP8AAAAAQAQAQAAZAAABQAAOkCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMMAAA
AAAAAIAAAQAAAAAIAAAUUGZFZABA5wDnBgQAAAAAXAQAIAAAABAAAAAIAAAQAAAA
EAAAABAAAAQAAAAEAAAABAAAAQAAAAAIAAAAwAAABQAAwABAAAAFAAEAC4AAAAEAQAQA
A5wb//wAA5wD//wAAAAEABAAAAEAAGADAAQABQAGAAcAAAAAAdGAKADIAhAEuAewCDAAAAEA
AAAAA2ED9AACAA3CQGeAsT9PAwB9AH0AAACAAAAAPHA/QAAwAHAAALIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAABAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgWBAgMFBQcICQkLCwwMDQ4NatoNDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA00Df0mDQ4NDawLCwkJCACFBQMCA239Jg4NDQ0LCwsJCQgHBQUDAgEBAgMFBQcICQkLCwsNDQ0
OAtO0DQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEvBT8LLwg3NT8INS8FNT8NNS8JBjU/BDUvCyMPAQytrQH5AgoEAQEBAArg
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAKGAQgEBAQMOBAkIBgcDAwEBAQEDAwMJAgE
BAxYLBQQEAWMCAgIEBAoBAQEECGcHBgUFBAMDAQEBAQFbWkFBQUGEf6tDwKEAwIBAQMDCGwVawc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBWgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXmZ8KNScxBxEfdjsSBHQEFDTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFeBAMDawIBAQGL7Y0EawQCAGIBAYYKChEQDQsJCACeBAU
CYt8BAQIDBAUFbQcHBwgICQgKjQECAGMEBAUFBgYHBgcIBWgCcaCHBwYGBgUFBQADAgIBAQEBAgI
DBAQFBQYGBgcHBwgmAQMDawUFBgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAgMDAwTV+94BAQECAwMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQEAWI
CUAgHBwcGBgYFBQQEAWMBAgIBAwMEBAUFbQcGBwCHCAImCAcHBwYGBgUFBQADAgIBAdUJCQgICAg
GBwYFBQADAgEBAAAAAIAAAAAA6cD9AADAawAADchNSElAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAIAAAEAAAABAAAAAABAAgAAQABAAAAAA
CAACACQABAAAAAADAAGAEAAABAAAAAEEAAGAGAABAAAAAFAAsAIAABAAAAAAGAAGAKwABAAA
AAAAKACwAMwABAAAAAALABIAxwADAAEECQAAAAIAcQADAAEECQABABAAcWADAAEECQACAA4AgwA
DAAEECQADABAAkQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lvbiAxLjB
idG4taWNvbkbZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBPAGMABwBuAFIAZQBnAHUAbABhAHIAyAgB0AG4ALQBPAGM
ABwBuAGIAAdABuAC0AAQBJAG8AbgBWAGUAcgBzAGkABwBuACAAMQAuADAAYgB0AG4ALQBPAGMABwB
uAEYABwBuAHQAIAABnAGUAbgBIAHIAyAgB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAEQBuAGMAZgB1AHM
AaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkABwB3AHcAdwAuAHMAEQBuAGMAZgB1AHMAaQB
vAG4ALgBjAG8AbQAAAAAIAAAIAAAoAAAAAIAAAIAAAIAAAIAAAIAAAIAAAIAAAIAAAIAAAIAAAIA
GAQcBCAEJAaptZWRpYS1wbGF5C211ZGlhLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AjbGlrc2S0
tLTAXBGNvcHkQLWRvd25sb2FkLTAYLXdmLQAA) format('trueType');
    font-weight: normal;
```

```

        font-style: normal;
    }
    .e-btn-sb-icon {
        font-family: 'btn-icon' !important;
        speak: none;
        font-size: 55px;
        font-style: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        line-height: 1;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
    }
    .e-download::before {
        content: '\e706';
    }
    .e-play::before {
        content: '\e700';
    }
    .e-pause::before {
        content: '\e701';
    }
    #container {
        visibility: hidden;
    }
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }

```

The method [dataBind](#) applies the property changes immediately to the component.

Start and stop methods

You can pause and resume the progress using the [stop](#) and [start](#) methods, respectively. In this demo, clicking the ProgressButton will pause and resume the progress.

INDEX.TS

```

import { ProgressButton } from '@syncfusion/ej2-splitbuttons';
let progressBtn: ProgressButton = new ProgressButton({
    content: 'Download', enableProgress: true, duration: 4000, iconCss: 'e-
btn-sb-icon e-download',
    end: () => {
        progressBtn.content = 'Download';
        progressBtn.iconCss = 'e-btn-sb-icon e-download';
        progressBtn.dataBind();
    },
    cssClass: 'e-hide-spinner'
});
progressBtn.appendTo('#progressbtn');
progressBtn.element.addEventListener('click', clickHandler);

```

```
function clickHandler(): void {
    if (progressBtn.content === 'Download') {
        progressBtn.content = 'Pause';
        progressBtn.iconCss = 'e-btn-sb-icon e-pause';
        progressBtn.dataBind();
    }
    // clicking on ProgressButton will stop the progress when the text
    content is 'Pause'
    else if (progressBtn.content === 'Pause') {
        progressBtn.content = 'Resume';
        progressBtn.iconCss = 'e-btn-sb-icon e-play';
        progressBtn.dataBind();
        progressBtn.stop();
    }
    // clicking on ProgressButton will start the progress when the text
    content is 'Resume'
    else if (progressBtn.content === 'Resume') {
        progressBtn.content = 'Pause';
        progressBtn.iconCss = 'e-btn-sb-icon e-pause';
        progressBtn.dataBind();
        progressBtn.start();
    }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ProgressButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="This demo for Essential JS2 Progress
button has progress indicator and spinner. It supports texts, icons, styles,
sizes, positions, and its customization.">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="progressbtn"></button>
  </div>
</body>
</html>
```



```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

@font-face {
    font-family: 'btn-icon';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXDNH+dzAAABoAAAAEJnbHlm1v4
8pAAAAfgAAQYAGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAArAAAACRobXR4IAAAAAAAYAAAAA
gbG9jYQON6ApQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYW1l07lFxAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGaABAAAEAAAAAFwEAAAAAAD9AABAAAAAIAAAACAAABAAAAQAAJ1LUzF8
PPPUACwQAAAAAANG+nFMAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAIAJ4AAwAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAIAAAAAAUGZFZABA5wDnBgQAAAAAXAQAABAAAAAABAAAAAQAQAAAA
EAAAABAAAAQAABAAAAAQAABAAAAAQAABAAAAAQAABAAAAAQAABAAAAAQAABAAAAAQAABAAAA
A5wb//wAA5wD//wAAAAEABAAAAEAAGADAAQABQAGAAcAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAA2ED9AACAA3CQGeAsT9PAwB9AH0AAACAAAAAPHa/QAAwAHAAAlIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgWBAgMFBQcICQkLCwwMDQ4NatoNDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA00df0mDQ4NDawLCwkJCACFBQMCA239Jg4NDQ0LCwsJCQgHBQUDAgEBAgMFBQcICQkLCwsNDQ0
OAtO0DQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDXQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEvBT8LLwg3NT8INS8FNT8NNS8JBjU/BDUvCyMPAQytrQH5AgoEAQEBAArg
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAjAgE
BAxYLBQQEAWMCAgIEBAoBAQEECGcHBgUFBAMDAQEBAQFbWkFBQUGEf6tDwKEAwIBAQMDCgVawc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAghCAcGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXmZ8KNScxBxEfdjSBHQEfDTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFeBAMDawIBAQGL7Y0EawQCAGIBAYYKChEQDQsJCACeBAU
CYt8BAQIDBAUFbQcHBwgICQgKjQECAGMEBAUFbGYHBGcIBwGcCAcHBwYGBgUFBQDAGIBAQEBAgI
DBAQFBQYGBGcHBwgmAQMDawUFbGYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAgMDAwTV+94BAQECAwMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQEAWI
CUAgHBwGBGyYFBQQAEMBAgIBAwMEBAUFbQcGBwCHCAImCAcHBwYGBgUFBQDAGIBAdUJCgICAg
GBwYFBQDAGEBAAAAAIAAAAAA6cD9AADAawAADchNSElAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAIAAAAAAABAAAAAABAAgAAQABAAAAA
CAACACQABAAAAAADAAGAEAAABAAAAAEEAGAGAABAAAAAFAAsAIAABAAAAAAGAAGAKwABAAA
AAAAKACwAMwABAAAAAALABIAXwADAAEECQAAAAIAcQADAAEECQABABAAcwADAAEECQACAA4AgwA
DAAEECQADABAAKQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bi1pY29uVmVyc2lvbiAxLjB
idG4taWNvbkbZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAyAgB0AG4ALQBpAGM
ABwBuAGIAdABuAC0AaQBJAG8AbgBWAGUAcgBzAGkABwBuACAAMQAuADAAYgB0AG4ALQBpAGMABwB
uAEYABwBuAHQAIAbnAGUAbgB1AHIAyAgB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAEQBuAGMAZgB1AHM
AaQBvAG4AIAABNAGUAdABYAG8AIABTahQAdQBkAGkABwB3AHcAdwAuAHMAEQBuAGMAZgB1AHMAaQB
vAG4ALgBjAG8AbQAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIA
GAQcBCAEJAaptZWRpYS1wbGF5C2l1ZGlhLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AjbGlrc2S0
tLTaxBGNvcHkQLWRvd25sb2FkLTAYLXdmLQAA) format('truetype');
    font-weight: normal;
    font-style: normal;
}

```

```

}
.e-btn-sb-icon {
    font-family: 'btn-icon' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-download::before {
    content: '\e706';
}
.e-play::before {
    content: '\e700';
}
.e-pause::before {
    content: '\e701';
}
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

See Also

- [How to hide spinner](#)
- [Customize ProgressBar using cssClass](#)

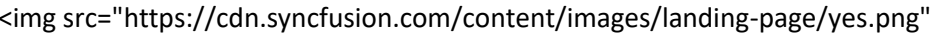
Accessibility in EJ2 JavaScript Progress button control

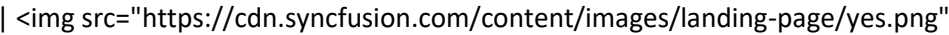
The Progress button component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Progress button component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" > |

| [Section 508](#) Support |  alt="Yes" > |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The
component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Progress button component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Progress button component:

Attributes	Purpose
aria-label	Provides an accessible name for the icon only Progress button.
aria-disabled	Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable.

Keyboard interaction

The Progress button component followed the [keyboard interaction] guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Progress button component.

| Press | To do this |

| --- | --- |

| Enter / Space | Starts the progress. |

Ensuring accessibility

The Progress button component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Progress button component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Progress button component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

How To

Change the text content and styles of the progressbutton during progress in EJ2 JavaScript Progress button control

You can change the text content and styles of the ProgressButton during progress by changing the text content and the [cssClass](#) property at the [begin](#) and [end](#) events.

INDEX.TS

```
import { ProgressButton } from '@syncfusion/ej2-splitbuttons';
let uploadBtn: ProgressButton = new ProgressButton({
  content: 'Upload',
  cssClass: 'e-hide-spinner',
  enableProgress: true,
  duration: 4000,
  begin: () => {
    uploadBtn.content = 'Uploading...';
    uploadBtn.cssClass = 'e-hide-spinner e-info';
    uploadBtn.dataBind();
  },
  end: () => {
    uploadBtn.content = 'Success';
    uploadBtn.cssClass = 'e-hide-spinner e-success';
    uploadBtn.dataBind();
    setTimeout(() => {
      uploadBtn.content = 'Upload';
      uploadBtn.cssClass = 'e-hide-spinner';
      uploadBtn.dataBind();
    }, 500)
  }
}, '#progressbtn');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ProgressButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="This demo for Essential JS2 Progress
button has progress indicator and spinner. It supports texts, icons, styles,
sizes, positions, and its customization.">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="progressbtn"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

@font-face {
    font-family: 'btn-icon';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXNlH+dzAAABoAAAAEJnbHlm1v4
8pAAAAfgAAQYAGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAAYAAAAA
gbG9jYQYQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYw1l07lFxAABhAAAAIxcG9zdK9uovo
AAAhEAAAAgAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAACAABAAAAQAAJ1LUzF8
PPPUACwQAAAAAANg+nFMAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAAlIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgWBAgMFBQcICQkLCwMDQ4NAtONDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA00Df0mDQ4NDawLCwkJCACFBQMCA239Jg4NDQ0LCwsJCQgHBQUDAgEBAgMFBQcICQkLCwsNDQ0
OAtO0DQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEvBT8LLwg3NT8INS8FNT8NNS8JByU/BDUvCyMPAQYtrQH5AgoEAQEBAQg
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAGABAQMOBAkIBgcDAwEBAQEDAwMJAgE

```

```

BAXYLBQQEAWMCAgIEBAoBAQEECgCHBgUFBAMDAQEBAQQFBwkFBQUGEf6tDwKEAwIBAQMDCgwVAwc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCakICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXMz8KNScxBxEfDjsBHQEfDTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQGL7Y0EAwQCAgIBAYYKChEQDQsJCAcEBAU
CYt8BAQIDBAUFBQcHBwgICQgKjQECAGMEBAUFBgYHBGcIBWgCcAcHBWYGBgUFBAQDAgIBAQEBAgI
DBAQFBQYGBGcHBwgmAQMDawUFBgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAGMDawTV+94BAQECawMDBAGyAQECBAYHCAGJCgkQCaQC6/47CQkICQcIBWYGBQQEAWI
CUAGHBwcGBgYFBQQEAWMBAgIBAwMEBAUFBQcGBwcHCAImCAcHBWYGBgUFBAQDAgIBAdUJCQgICAg
GBWYFBQDAgEBAAAAAIAAAAAA6cD9AADAawAADchNSELAQcJAScBESNZAO78sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAEEEEAAAAAABAAAAAABAAgAAQABAAAAA
CAACACQABAAAAAADAAGAEABAAAAAEEAAGAGAABAAAAAFAAAsAIAABAAAAAAGAAgAKWABAAAA
AAAAKACwAMwABAAAAAALABIAxwADAAEECQAAAAIacQADAAEECQABABAAcwADAAEECQACAA4AgwA
DAAEECQADABAAKQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lvbiAxLjB
idG4taWNvbkbZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAAYgB0AG4ALQBpAGM
ABwBuAGIAAdABuAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMABwB
uAEYABwBuAHQAIAbnAGUAbgBlAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBuaGMAZgB1AHM
AaQBvAG4AIAIBNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuaGMAZgB1AHMAaQB
vAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAAAGBAgEDAQQBBQE
GAQCBCAEJAaptZWRpYS1wbGF5C21lZG1hLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AjbGlrc2S0
tLTAXBGNvcHkQLWRvd25sb2FkLTAYLXdmlQAA) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-btn-sb-icon {
    font-family: 'btn-icon' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-download::before {
    content: '\e706';
}
.e-play::before {
    content: '\e700';
}
.e-pause::before {
    content: '\e701';
}
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
}

```

```
width: 30%;
}
```

Customize progress using cssclass in EJ2 JavaScript Progress button control

You can customize the background filler UI using the [cssClass](#) property.

- Adding `e-vertical` to `cssClass` shows vertical progress.
- Adding `e-progress-top` to `cssClass` shows progress at the top.

You can also show reverse progress by adding custom class to the [cssClass](#) property.

INDEX.TS

```
import { ProgressButton } from '@syncfusion/ej2-splitbuttons';
let verticalProgress: ProgressButton = new ProgressButton({ content:
'Vertical Progress', enableProgress: true, cssClass: 'e-hide-spinner e-
vertical', duration: 4000 });
verticalProgress.appendTo('#vertical');
let topProgress: ProgressButton = new ProgressButton({ content: 'Progress
Top', enableProgress: true, cssClass: 'e-hide-spinner e-progress-top',
duration: 4000 });
topProgress.appendTo('#top');
let reverseProgress: ProgressButton = new ProgressButton({ content: 'Reverse
Progress', enableProgress: true, cssClass: 'e-hide-spinner e-reverse-
progress', duration: 4000 });
reverseProgress.appendTo('#reverse');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ProgressButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="This sample is to customize progress
indicator with top, reverse and vertical in the progress button">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

    <div id="container">
      <button id="vertical"></button>
      <button id="top"></button>
      <button id="reverse"></button>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

.e-reverse-progress .e-progress {
  right: 0;
  left: auto;
}
button {
  margin: 25px;
}
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}

```

Hide spinner in EJ2 JavaScript Progress button control

You can hide spinner in the ProgressButton by setting the `e-hide-spinner` property to [cssClass](#).

INDEX.TS

```

import { ProgressButton } from '@syncfusion/ej2-splitbuttons';
let progressBtn: ProgressButton = new ProgressButton({ content: 'Progress',
enableProgress: true, cssClass: 'e-hide-spinner' });
progressBtn.appendTo('#progressbtn');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 ProgressButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="This demo for Essential JS2 Progress
button has progress indicator and spinner. It supports texts, icons, styles,
sizes, positions, and its customization.">
  <meta name="author" content="Syncfusion">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="progressbtn"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

@font-face {
    font-family: 'btn-icon';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXNlH+dzAAABoAAAAEJnbHlm1v4
8pAAAAfgAAAQYaGVhZBOPfZcAAADQAAANmhoZWEIUQQJAAArAAAACRobXR4IAAAAAAAAAAAAA
gbG9jYQn6ApQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYw1l07lFxAAABhAAAAIXcG9zdK9uovo
AAAhEAAAAgAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAABAAAAAAQAAJ1LUzF8
PPPUACwQAAAAAANg+nFMAAAAA2D6cUwAAAAAD9AP0AAAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAAAAABAAAAAAAAAAAAAQA
AAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAEAAgADAAQABQAGAAcAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAALIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgwBAGMFBQCICQkLCwwMDQ4NAtONDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA00Df0mDQ4NDawLCwkJCAcFBQMCA239Jg4NDQ0LCwsJCQgHBQUdAgEBAgMFBQCICQkLCwsNDQ0
OAtOoDQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEvBT8LLwg3NT8INS8FNT8NNS8JByU/BDUvCyMPAQytrQH5AgoEAQEBAArg
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAGABAQMOBAkIBgcDAwEBAQEDAwMJAgE
BAxYLBQQEAWMCAGIEBAoBAQEECgcHBgUFBAMDAQEBAQQFBwkFBQUGef6tDwKEAwIBAQMDCGwVAwC
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQCGBwgHBgY
GBgoJCAYCAGABAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM

```

```

hLwIDNzM/CjUTHwcVIwcVIy8HETcXMz8KNScxBxEfDjsBHQEfDTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQL7Y0EawQCAgIBAYYKChEQDQsJCAcEBAU
CYt8BAQIDBAUFbQcHBwgICQgKjQECAGMEBAUFbgYHBgcIBWgCcAcHBwYGBgUFBAQDAgIBAQEBAgI
DBAQFBQYGBgcHBwgmaQMDAwUFbgYHBwgICQkJ/tQCiwMEBf3XAwYEAgiEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAGMDAwTV+94BAQECAWMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQEAwI
CUAGHBwcGBgYFBQQEAwMBAgIBAwMEBAUFbQcGBwCHCAImCAcHBwYGBgUFBAQDAgIBAdUJCQgICAg
GBwYFBQAQDAgEBAAAAAIAAAAAA6cD9AADAaWaadchNSElAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EWMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAEEEEAAAAAABAAAAAABAAgAAQABAAAAA
CAACACQABAAAAAADAAGAEABAAAAAEEAGAGAABAAAAAFAAsAIAABAAAAAAGAAgAKwABAAA
AAAAKACwAMwABAAAAAALABIAXwADAAEECQAAAAIACQADAAEECQABABAAcwADAAEECQACAA4AgwA
DAAEECQADABAAKQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lvbiAxLjB
idG4taWNvbkZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMAbwBuAFIAZQBnAHUAbABhAHIAyB0AG4ALQBpAGM
AbwBuAGIAAdABuAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMAbwB
uAEYAbwBuAHQAIABnAGUAbgBlAHIAyQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBuaGMAZgB1AHM
AaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuaGMAZgB1AHMAaQB
vAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
GAQcBCAEJAAPtZWRpYS1wbGF5C21lZGlhLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrZS0
tLTaxBGNvcHkQLWRvd25sb2FkLTAYLXdmlQAQ) format('true');
    font-weight: normal;
    font-style: normal;
}
.e-btn-sb-icon {
    font-family: 'btn-icon' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-download::before {
    content: '\e706';
}
.e-play::before {
    content: '\e700';
}
.e-pause::before {
    content: '\e701';
}
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Trace events of progress button in EJ2 JavaScript Progress button control

The ProgressButton component triggers events based on its actions. The events can be used as extension points to perform custom operations.

The events available in ProgressButton are [fail](#), [begin](#), [progress](#), and [end](#).

INDEX.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { ProgressButton, ProgressEventArgs } from '@syncfusion/ej2-splitbuttons';
let progressBtn: ProgressButton = new ProgressButton({
    content: 'Progress',
    enableProgress: true,
    begin: (args: ProgressEventArgs) => {
        updateEventLog(args);
    },
    end: (args: ProgressEventArgs) => {
        updateEventLog(args);
    },
    progress: (args: ProgressEventArgs) => {
        updateEventLog(args);
    },
    fail: (args: Event) => {
        updateEventLog(args);
    }
}, '#progressbtn');
let clear: Button = new Button({ cssClass: 'e-small' });
clear.appendTo('#clear');
clear.element.onclick = () => {
    let propertyElem: HTMLElement =
document.getElementById('propertyTable');
    propertyElem.getElementsByTagName('td')[0].innerHTML = '';
}
function updateEventLog(args: any): void {
    let propertyElem: HTMLElement =
document.getElementById('propertyTable');

propertyElem.getElementsByTagName('td')[0].insertAdjacentHTML('beforeend',
args.name + ' Event triggered. <br />');
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 ProgressButton</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="This sample is to customize progress
indicator with top, reverse and vertical in the progress button">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="control-section">
            <div class="progress-btn-section">
                <button id="progressbtn"></button>
            </div>
            <div class="property-section">
                <table id="propertyTable" title="Event trace">
                    <tbody>
                        <tr><th>Event trace:-</th>
                        </tr><tr>
                            <td></td>
                        </tr>
                    </tbody>
                </table>
            </div>
            <button id="clear">Clear</button>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

html, body, .control-section {
    height: 95%;
}
.progress-btn-section {
    text-align: center;
    float: left;
}
.property-section {
    overflow: auto;
    width: 40%;
    height: 330px;
    float: right;
    font-family: monospace;
}

```

```

}
.property-section th {
    text-align: left;
}
#clear {
    float: right;
    clear: both;
}

```

Query Builder

Columns in EJ2 JavaScript Query builder control

The column definitions are used as the [dataSource](#) schema in the Query Builder. This plays a vital role in rendering column values. The query builder operations such as create or delete conditions, and create or delete groups, are performed based on the column definitions. The [field](#) property of the [columns](#) is necessary to map the data source values in the query builder columns.

If the column field is not specified in the dataSource, the column values will be empty.

Auto generation

The [columns](#) are automatically generated when the [columns](#) declaration is empty or undefined while initializing the query builder. All the columns in the [dataSource](#) are bound as the query builder columns.

INDEX.TS

```

import { QueryBuilder } from '@syncfusion/ej2-querybuilder';
let data: Object[] = [
    { OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5, Freight: 32.3800,
      OrderDate: "07/09/1991" },
    { OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6, Freight: 32.3800,
      OrderDate: "07/09/1991" },
    { OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4, Freight: 32.3800,
      OrderDate: "07/09/1991" }];
let qryBldrObj: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: data,
});
qryBldrObj.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div id="property"> </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

When columns are auto-generated, the column type will be determined from the first record of the dataSource.

Labels

By default, the column label is displayed from the column [field](#) value. To override the default label, you have to define the [label](#) value.

Operators

The operator for a column can be defined in the [operators](#) property.

The available operators and its supported data types are:

Operators	Description	Supported Types
-----	-----	-----
startswith	Checks whether the value begins with the specified value.	String
endswith	Checks whether the value ends with the specified value.	String
contains	Checks whether the value contains the specified value.	String

| equal | Checks whether the value is equal to the specified value. | String Number Date Boolean |

| notequal | Checks whether the value is not equal to the specified value. | String Number Date Boolean |

| greaterthan | Checks whether the value is greater than the specified value. | Date Number |

| greaterthanorequal | Checks whether a value is greater than or equal to the specified value. | Date Number |

| lessthan | Checks whether the value is less than the specified value. | Date Number |

| lessthanorequal | Checks whether the value is less than or equal to the specified value. | Date Number |

| between | Checks whether the value is between the two-specific values. | Date Number |

| notbetween | Checks whether the value is not between the two-specific values. | Date Number |

| in | Checks whether the value is one of the specific values. | String Number |

| notin | Checks whether the value is not in the specific values. | String Number |

Step

The Query Builder allows you to set the step values to the number fields. So that, you can easily access the numeric textbox. Use the [step](#) property, to set the step value for number values.

Format

The Query Builder formats date and number values. Use the [format](#) property to format date and number values.

INDEX.TS

```
import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
let employeeData: Object[] = [{
    'EmployeeID': 1,
    'FirstName': 'Nancy',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
    'City': 'Seattle',
    'Country': 'USA'
},
{
    'EmployeeID': 2,
    'FirstName': 'Andrew',
    'Title': 'Vice President',
    'TitleOfCourtesy': 'Dr.',
    'HireDate': '21/04/2003',
    'City': 'Tacoma',
    'Country': 'USA'
},
{
    'EmployeeID': 3,
    'FirstName': 'Janet',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
```

```

        'City': 'Kirkland',
        'Country': 'USA'
    }];
    let columnData: ColumnsModel[] = [
        {
            field: 'EmployeeID', label: 'EmployeeID', type: 'number',
            operators: [{ key: 'Equal', value: 'equal' },
                { key: 'Greater than', value: 'greaterthan' }, { key: 'Less
than', value: 'lessthan' }], step: 10
        },
        { field: 'FirstName', label: 'FirstName', type: 'string' },
        { field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type:
'boolean', values: ['Mr.', 'Mrs.'] },
        { field: 'Title', label: 'Title', type: 'string' },
        { field: 'HireDate', label: 'HireDate', type: 'date', format:
'dd/MM/yyyy' },
        { field: 'Country', label: 'Country', type: 'string' },
        { field: 'City', label: 'City', type: 'string' }
    ];
    let importRules: RuleModel = {
        'condition': 'and',
        'rules': [{
            'label': 'HireDate',
            'field': 'HireDate',
            'type': 'date',
            'operator': 'equal',
            'value': '07/05/1991'
        },
        {
            'label': 'Title',
            'field': 'Title',
            'type': 'string',
            'operator': 'equal',
            'value': 'Sales Manager'
        }
    ]
    };
    let qryBldrObj: QueryBuilder = new QueryBuilder({
        width: '70%',
        dataSource: employeeData,
        columns: columnData,
        rule: importRules,
    });
    qryBldrObj.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-querybuilder/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div id="property"> </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Validation

Validation allows you to validate the conditions and it display errors for invalid fields while using the `validateFields` method. To enable validation in the query builder , set the `allowValidation` to true. Column fields are validated after setting [allowValidation](#) to true. So, you should manually configure the validation for Operator and Value fields through [validation](#).

Set [isRequired](#) validation for Operator and Value fields.

Set [min](#), [max](#) values for number values.

INDEX.TS

```

import { QueryBuilder, ColumnsModel } from '@syncfusion/ej2-querybuilder';
import { Button } from '@syncfusion/ej2-buttons';
let employeeData: Object[] = [{
    'EmployeeID': 1,
    'FirstName': 'Nancy',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',

```

```

        'HireDate': '22/07/2001',
        'City': 'Seattle',
        'Country': 'USA'
    },
    {
        'EmployeeID': 2,
        'FirstName': 'Andrew',
        'Title': 'Vice President',
        'TitleOfCourtesy': 'Dr.',
        'HireDate': '21/04/2003',
        'City': 'Tacoma',
        'Country': 'USA'
    },
    {
        'EmployeeID': 3,
        'FirstName': 'Janet',
        'Title': 'Sales Representative',
        'TitleOfCourtesy': 'Ms.',
        'HireDate': '22/07/2001',
        'City': 'Kirkland',
        'Country': 'USA'
    }
    ];
    let columnData: ColumnsModel[] = [
        { field: 'EmployeeID', label: 'EmployeeID', type: 'number',
validation: { isRequired: true } },
        { field: 'FirstName', label: 'FirstName', type: 'string' },
        { field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type:
'boolean', values: ['Mr.', 'Mrs.'] },
        { field: 'Title', label: 'Title', type: 'string' , validation: {
isRequired: true } },
        { field: 'HireDate', label: 'HireDate', type: 'date', format:
'dd/MM/yyyy' },
        { field: 'Country', label: 'Country', type: 'string', validation: {
isRequired: true } },
        { field: 'City', label: 'City', type: 'string', validation: {
isRequired: true } }
    ];
    let qryBldrObj: QueryBuilder = new QueryBuilder({
        width: '70%',
        dataSource: employeeData,
        columns: columnData,
        allowValidation: true
    });
    qryBldrObj.appendTo('#querybuilder');
    let button: Button = new Button({cssClass: `e-primary`,
content: 'Validation'}, '#qbbbutton');
    document.getElementById('qbbbutton').onclick = (): void => {
        qryBldrObj.validateFields();
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div class="e-qb-button">
            <button id="qbbutton" class="e-btn e-primary"> Validate Fields
        </button>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Data binding in EJ2 JavaScript Query builder control

The Query Builder uses **DataManager** to bind the datasource, which supports both RESTful JSON data services binding and local JavaScript object array binding. The [dataSource](#) property can be assigned either with the instance of **DataManager** or JavaScript object array collection. It supports two kind of databinding method:

- Local data
- Remote data

Local data

To bind local data to the query builder, you can assign the [dataSource](#) property with a JavaScript object array. The local data source can also be provided as an instance of the [DataManager](#).

INDEX.TS

```
import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
let employeeData: Object[] = [{
    'EmployeeID': 1,
    'FirstName': 'Nancy',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
    'City': 'Seattle',
    'Country': 'USA'
},
{
    'EmployeeID': 2,
    'FirstName': 'Andrew',
    'Title': 'Vice President',
    'TitleOfCourtesy': 'Dr.',
    'HireDate': '21/04/2003',
    'City': 'Tacoma',
    'Country': 'USA'
},
{
    'EmployeeID': 3,
    'FirstName': 'Janet',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
    'City': 'Kirkland',
    'Country': 'USA'
}
];
let columnData: ColumnsModel[] = [
    {
        field: 'EmployeeID', label: 'EmployeeID', type: 'number'
    },
    { field: 'FirstName', label: 'FirstName', type: 'string' },
    { field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type: 'boolean', values: ['Mr.', 'Mrs.'] },
    { field: 'Title', label: 'Title', type: 'string' },
    { field: 'HireDate', label: 'HireDate', type: 'date', format: 'dd/MM/yyyy' },
    { field: 'Country', label: 'Country', type: 'string' },
    { field: 'City', label: 'City', type: 'string' }
];
let importRules: RuleModel = {
    'condition': 'and',
    'rules': [{
        'label': 'EmployeeID',
        'field': 'EmployeeID',
        'type': 'number',
        'operator': 'equal',
        'value': 1001
    }
],
}
```

```

        {
            'label': 'Title',
            'field': 'Title',
            'type': 'string',
            'operator': 'equal',
            'value': 'Sales Manager'
        }
    ];
    let qryBldrObj: QueryBuilder = new QueryBuilder({
        width: '70%',
        dataSource: employeeData,
        columns: columnData,
        rule: importRules,
    });
    qryBldrObj.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="querybuilder"></div>
    <div id="property"> </div>
  </div>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, **DataManager** uses **JsonAdaptor** for local data-binding.

Remote data

To bind remote data to the query builder, assign service data as an instance of **DataManager** to the [dataSource](#) property. To interact with remote data source, provide the endpoint **url**.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
  url: 'https://services.syncfusion.com/js/production/api/employees',
  adaptor: new ODataAdaptor()
});

let columnData: ColumnsModel[] = [
  { field: 'EmployeeID', label: 'EmployeeID', type: 'number' },
  { field: 'FirstName', label: 'FirstName', type: 'string' },
  { field: 'Designation', label: 'Designation', type: 'string' },
  { field: 'Country', label: 'Country', type: 'string' }
];

let importRules: RuleModel = {
  'condition': 'and',
  'rules': [{
    'label': 'EmployeeID',
    'field': 'EmployeeID',
    'type': 'number',
    'operator': 'equal',
    'value': 1001
  },
  {
    'label': 'Country',
    'field': 'Country',
    'type': 'string',
    'operator': 'equal',
    'value': 'USA'
  }
]
};

let qryBldrObj: QueryBuilder = new QueryBuilder({
  width: '70%',
  dataSource: data,
  columns: columnData,
  rule: importRules,
});
qryBldrObj.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="querybuilder"></div>
    <div id="property"> </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, **DataManager** uses **ODataAdaptor** for remote data-binding.

Binding with OData services

OData is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the DataManager. Refer to the following code example for remote Data binding using OData service.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://services.syncfusion.com/js/production/api/Employees/',
    adaptor: new ODataAdaptor(),
    crossDomain: true
});

let columnData: ColumnsModel[] = [
    {
        field: 'EmployeeID', label: 'EmployeeID', type: 'number'
    },
    { field: 'FirstName', label: 'FirstName', type: 'string' },
    { field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type: 'boolean', values: ['Mr.', 'Mrs.'] },
    { field: 'Title', label: 'Title', type: 'string' },
    { field: 'HireDate', label: 'HireDate', type: 'date', format: 'dd/MM/yyyy' },
    { field: 'Country', label: 'Country', type: 'string' },
    { field: 'City', label: 'City', type: 'string' }
];

let importRules: RuleModel = {
    'condition': 'and',
    'rules': [{
        'label': 'EmployeeID',
        'field': 'EmployeeID',
        'type': 'number',
        'operator': 'equal',
        'value': 1001
    },
    {
        'label': 'Title',
        'field': 'Title',
        'type': 'string',
        'operator': 'equal',
        'value': 'Sales Manager'
    }
    ]
};

let qryBldrObj: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: data,
    columns: columnData,
    rule: importRules,
});

qryBldrObj.appendTo('#querybuilder');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Query Builder</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div id="property"> </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Binding with OData v4 services

The ODataV4 is an improved version of OData protocols, and the **DataManager** can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the [odata documentation](#). To bind OData v4 service, use the **ODataV4Adaptor**.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-
querybuilder';
import { DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Employees/',
    adaptor: new ODataV4Adaptor()
});
let columnData: ColumnsModel[] = [

```

```

        {
            field: 'EmployeeID', label: 'EmployeeID', type: 'number'
        },
        { field: 'FirstName', label: 'FirstName', type: 'string' },
        { field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type:
'boolean', values: ['Mr.', 'Mrs.'] },
        { field: 'Title', label: 'Title', type: 'string' },
        { field: 'HireDate', label: 'HireDate', type: 'date', format:
'dd/MM/yyyy' },
        { field: 'Country', label: 'Country', type: 'string' },
        { field: 'City', label: 'City', type: 'string' }
    ];
    let importRules: RuleModel = {
        'condition': 'and',
        'rules': [{
            'label': 'EmployeeID',
            'field': 'EmployeeID',
            'type': 'number',
            'operator': 'equal',
            'value': 1001
        },
        {
            'label': 'Title',
            'field': 'Title',
            'type': 'string',
            'operator': 'equal',
            'value': 'Sales Manager'
        }
    ]
    };
    let qryBldrObj: QueryBuilder = new QueryBuilder({
        width: '70%',
        dataSource: data,
        columns: columnData,
        rule: importRules,
    });
    qryBldrObj.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Query Builder</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div id="property"> </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Web API

You can use **WebApiAdaptor** to bind query builder with Web API created using OData endpoint.

`ts

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';

let data: DataManager = new DataManager({
    url: '/api/OrderAPI',
    adaptor: new WebApiAdaptor
});

let columnData: ColumnsModel[] = [
    {
        field: 'EmployeeID', label: 'EmployeeID', type: 'number'
    },
    { field: 'FirstName', label: 'FirstName', type: 'string' },
    { field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type: 'boolean', values: ['Mr.', 'Mrs.'] },

```

```
{ field: 'Title', label: 'Title', type: 'string' },
{ field: 'HireDate', label: 'HireDate', type: 'date', format: 'dd/MM/yyyy' },
{ field: 'Country', label: 'Country', type: 'string' },
{ field: 'City', label: 'City', type: 'string' }
];
let importRules: RuleModel = {
  'condition': 'and',
  'rules': [{
    'label': 'EmployeeID',
    'field': 'EmployeeID',
    'type': 'number',
    'operator': 'equal',
    'value': 1001
  },
  {
    'label': 'Title',
    'field': 'Title',
    'type': 'string',
    'operator': 'equal',
    'value': 'Sales Manager'
  }
  ]
};
let qryBldrObj: QueryBuilder = new QueryBuilder({
  width: '70%',
  dataSource: data,
  columns: columnData,
  rule: importRules,
});
qryBldrObj.appendTo('#querybuilder');
```

Support with Data Manager

You can use the created conditions in DataManager through the [getPredicate](#) method, which results the filtered records.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-
querybuilder';
import { Button } from '@syncfusion/ej2-buttons';
import { DataManager, Query } from '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let hardwareData: Object[] = [{
    'TaskID': 1,
    'Name': 'Lenovo Yoga',
    'Category': 'Laptop',
    'SerialNo': 'CB27932009',
    'InvoiceNo': 'INV-2878',
    'Status': 'Assigned'
},
{
    'TaskID': 2,
    'Name': 'Acer Aspire',
    'Category': 'Others',
    'SerialNo': 'CB35728290',
    'InvoiceNo': 'INV-3456',
    'Status': 'In-repair'
},
{
    'TaskID': 3,
    'Name': 'Apple MacBook',
    'Category': 'Laptop',
    'SerialNo': 'CB35628728',
    'InvoiceNo': 'INV-2763',
    'Status': 'In-repair'
}
]];
let columnData: ColumnsModel[] = [
    { field: 'TaskID', label: 'Task ID', type: 'number' },
    { field: 'Name', label: 'Name', type: 'string' },
    { field: 'Category', label: 'Category', type: 'string' },
    { field: 'SerialNo', label: 'Serial No', type: 'string' },
    { field: 'InvoiceNo', label: 'Invoice No', type: 'string' },
    { field: 'Status', label: 'Status', type: 'string' }
];
let importRules: RuleModel = {
    'condition': 'and',
    'rules': [{
        'label': 'Task ID',
        'field': 'TaskID',
        'type': 'number',
        'operator': 'equal',
        'value': 1
    }
    ]
};
let qryBldrObj: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: hardwareData,
    columns: columnData,
    rule: importRules,
});
qryBldrObj.appendTo('#querybuilder');
let template: string =
'<tr><td>${TaskID}</td><td>${Category}</td><td>${Status}</td></tr>';
let dataManagerQuery: Query;

```

```

let button: Button = new Button({cssClass: `e-primary`, content: 'get
data'}, '#getdata');
let validRule: RuleModel = qryBldrObj.getValidRules(qryBldrObj.rule);
document.getElementById('getdata').onclick = (): void => {
    let compiledFunction: Function = compile(template);
    document.getElementById('datatable').style.display = 'block';
    dataManagerQuery = new Query().select(['TaskID', 'Category',
'Status']).where(qryBldrObj.getPredicate(validRule)).take(8);
    let result: Object[] = new
DataManager(hardwareData).executeLocal(dataManagerQuery);
    let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
    if (table.querySelectorAll('tbody')[1]) {
        table.querySelectorAll('tbody')[1].remove();
    }
    result.forEach((data: Object) => {
        table.appendChild(compiledFunction(data)[0]);
    });
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Query Builder</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

<div id="querybuilder"></div>
  <div class="e-qb-button">
    <button id="getdata" class="e-btn e-primary">get Data</button>
  </div>
  <div id="Grid"></div>
  <table id="datatable" class="e-table" style="display:none">
    <thead>
      <tr><th>TaskID</th><th>Category</th><th>Status</th></tr>
    </thead>
    <tbody>
    </tbody>
  </table>
</div>

<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Complex Data Binding

Complex Data Binding allows you to create subfield for columns. To implement complex data binding, either bind the complex data in nested columns or specify complex data source and separator must be given in querybuilder.

In the following sample, complex data was bound in nested columns.

INDEX.TS

```

import { QueryBuilder, RuleModel, RuleChangeEventArgs, ColumnsModel } from '@syncfusion/ej2-querybuilder';
import { addClass, removeClass, getComponent } from '@syncfusion/ej2-base';
import { RadioButton, ChangeEventArgs } from '@syncfusion/ej2-buttons';
import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
let importRules: RuleModel = {
  condition: 'and',
  rules: [{
    label: 'ID',
    field: 'Employee.ID',
    type: 'string',
    operator: 'equal',
    value: 0
  },
  {
    label: 'Last Name',
    field: 'Name.LastName',
    type: 'string',
    operator: 'contains',
    value: 'malan'
  }
],
  condition: 'or',
  rules: [{

```

```

        label: 'City',
        field: 'Country.State.City',
        operator: 'startswith',
        type: 'string',
        value: 'U'
    }, {
        label: 'Region',
        field: 'Country.Region',
        operator: 'endswith',
        type: 'string',
        value: 'c'
    }, {
        label: 'Name',
        field: 'Country.Name',
        operator: 'isnotempty'
    }
    ]
}
});
let columns: ColumnsModel[] = [
    {field: 'Employee', label: 'Employee', columns: [
        { field: 'ID', label: 'ID', type: 'number'},
        { field: 'DOB', label: 'Date of birth', type: 'date'},
        { field: 'HireDate', label: 'Hire Date', type: 'date'},
        { field: 'Salary', label: 'Salary', type: 'number'},
        { field: 'Age', label: 'Age', type: 'number'},
        { field: 'Title', label: 'Title', type: 'string'}
    ]},
    {field: 'Name', label: 'Name', columns: [
        { field: 'FirstName', label: 'First Name', type: 'string'},
        { field: 'LastName', label: 'Last Name', type: 'string'}
    ]},
    {field: 'Country', label: 'Country', columns : [
        { field: 'State', label: 'State', columns : [
            { field: 'City', label: 'City', type: 'string'},
            { field: 'Zipcode', label: 'Zip Code', type: 'number'}] },
        { field: 'Region', label: 'Region', type: 'string'},
        { field: 'Name', label: 'Name', type: 'string'}
    ]}
];
let qryBldrObj: QueryBuilder = new QueryBuilder({
    separator: '.',
    columns: columns,
    rule: importRules,
    enableNotCondition: true
});
qryBldrObj.appendTo('#querybuilder');
document.getElementById('rule').onclick = (e: Event) => {
    qryBldrObj.setRulesFromSql("Employee.ID = 0 AND Name.LastName LIKE ('%malan%') AND (Country.State.City LIKE ('U%') AND Country.Region LIKE ('%c')) AND Country.Name IS NOT EMPTY");
}
document.getElementById('sql').onclick = (e: Event) => {
    qryBldrObj.setRules(importRules);
}
document.getElementById('reset').onclick = (e: Event) => {
    qryBldrObj.reset();
}
}

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <table>
        <tbody><tr>
          <td> <button id="reset" class="e-control e-danger e-btn
e-small">Reset</button> </td>
          <td> <button id="rule" class="e-control e-success e-btn
e-small">SetSqlRules</button> </td>
          <td> <button id="sql" class="e-control e-success e-btn
e-small">SetRules</button> </td>
        </tr>
      </tbody></table>
    </div>
    <div class="col-lg-8 control-section">
      <div class="querybuilder-main">
        <div id="querybuilder">
        </div>
      </div>
    </div>
  </div>

```

```

<style type="text/css">
.e-query-builder .e-group-body .e-horizontal-mode .e-rule-sub-filter{
    display: inline-block;
}
.e-query-builder .e-group-body .e-rule-container .e-rule-sub-filter{
    padding: 12px 0 12px 12px;
    width: auto;
}
</style>
</div><script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filtering in EJ2 JavaScript Query builder control

Query Builder allows you to create or delete conditions and groups. You can use [showButtons](#) to enable/disable these buttons.

You can create or delete conditions by interacting through the user interface and methods.

- Use the [addRules](#), and [deleteRules](#) methods to create/delete conditions.
- Use [addGroups](#), and [deleteGroups](#) methods to create/delete groups.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
import { Button } from '@syncfusion/ej2-buttons';
let employeeData: Object[] = [{
    'EmployeeID': 1,
    'FirstName': 'Nancy',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
    'City': 'Seattle',
    'Country': 'USA'
},
{
    'EmployeeID': 2,
    'FirstName': 'Andrew',
    'Title': 'Vice President',
    'TitleOfCourtesy': 'Dr.',
    'HireDate': '21/04/2003',
    'City': 'Tacoma',
    'Country': 'USA'
},
{
    'EmployeeID': 3,
    'FirstName': 'Janet',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',

```

```

        'HireDate': '22/07/2001',
        'City': 'Kirkland',
        'Country': 'USA'
    }];
    let columnData: ColumnsModel[] = [
        {
            field: 'EmployeeID', label: 'Employee ID', type: 'number'
        },
        { field: 'FirstName', label: 'First Name', type: 'string' },
        { field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type:
'boolean', values: ['Mr.', 'Mrs.'] },
        { field: 'Title', label: 'Title', type: 'string' },
        { field: 'HireDate', label: 'Hire Date', type: 'date', format:
'dd/MM/yyyy' },
        { field: 'Country', label: 'Country', type: 'string' },
        { field: 'City', label: 'City', type: 'string' }
    ];
    let importRules: RuleModel = {
        'condition': 'and',
        'rules': [{
            'label': 'Employee ID',
            'field': 'EmployeeID',
            'type': 'number',
            'operator': 'equal',
            'value': 1001
        },
        {
            'label': 'Title',
            'field': 'Title',
            'type': 'string',
            'operator': 'equal',
            'value': 'Sales Manager'
        }
    ]
    };
    let qryBldrObj: QueryBuilder = new QueryBuilder({
        width: '70%',
        dataSource: employeeData,
        columns: columnData,
        showButtons: { ruleDelete: true , groupInsert: true, groupDelete:
true }
        rule: importRules
    });
    qryBldrObj.appendTo('#querybuilder');
    let button: Button = new Button({cssClass: `e-primary`, content: 'Add
Rule' }, '#qbaddrule');
    button = new Button({cssClass: `e-primary`, content: 'Add Group'},
'#qbaddgroup');
    button = new Button({cssClass: `e-primary`, content: 'Delete Group' },
'#qbdeletegroup');
    document.getElementById('qbaddrule').onclick = (): void => {
        qryBldrObj.addRules([{ 'label': 'Employee ID', 'field':
'EmployeeID', 'type': 'number', 'operator': 'equal', 'value': '1091' }],
'group0');
    }
    document.getElementById('qbaddgroup').onclick = (): void => {

```

```

    gryBldrObj.addGroups([{'condition': 'and', 'rules': [{ 'label': 'First
Name', 'field': 'FirstName', 'type': 'string', 'operator':
'startswith', 'value': 'v' } ]}], 'group0');
  }
  document.getElementById('qbdeletegroup').onclick = (): void => {
    gryBldrObj.deleteGroups(['group1']);
  }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="querybuilder"></div>
    <div class="e-qb-button">
      <button id="qbaddrule" class="e-btn e-primary"> Add Rule
</button>
      <button id="qbaddgroup" class="e-btn e-primary"> Add Group
</button>
      <button id="qbdeletegroup" class="e-btn e-primary">Delete Group
</button>
    </div>
  </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Model binding in EJ2 JavaScript Query builder control

Model binding allows to bind properties for the components used in field, operator, and value columns. To implement model binding, assign fieldModel, operatorModel, and valueModel properties in QueryBuilder.

INDEX.TS

```

import { QueryBuilder, RuleModel, ColumnsModel } from '@syncfusion/ej2-querybuilder';
import { getComponent, extend } from '@syncfusion/ej2-base';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
let filter: ColumnsModel [] = [
    { field: 'EmployeeID', label: 'EmployeeID', type: 'number' },
    { field: 'FirstName', label: 'FirstName', type: 'string' },
    { field: 'LastName', label: 'LastName', type: 'string' },
    { field: 'Age', label: 'Age', type: 'number' },
    { field: 'City', label: 'City', type: 'string' },
    { field: 'Country', label: 'Country', type: 'string' },
];
let importRules: RuleModel = {
    'condition': 'and',
    'rules': [{
        'label': 'Employee ID',
        'field': 'EmployeeID',
        'type': 'number',
        'operator': 'equal',
        'value': 1001
    }]
};
let qryBldrObj: QueryBuilder = new QueryBuilder({
    columns: filter,
    rule: importRules,
    enableNotCondition: true,
    displayMode: "Horizontal",
    allowValidation: true,
    fieldModel: {
        allowFiltering: true,
        popupHeight: '400px'
    },
    operatorModel: {
        allowFiltering: true,
        popupHeight: '400px'
    },
    valueModel: {
        numericTextBoxModel: {
            cssClass: 'e-custom'
        },
        multiSelectModel: {

```

```

        cssClass: 'e-custom'
    },
    datePickerModel: {
        cssClass: 'e-custom'
    },
    textBoxModel: {
        cssClass: 'e-custom'
    },
    radioButtonModel: {
        cssClass: 'e-custom'
    }
}
});
qryBldrObj.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="querybuilder"></div>
  </div>
  <style type="text/css">
</style>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Templates in EJ2 JavaScript Query builder control

Templates allows users to define customized header and own user interface for columns.

Header Template

Header Template allows to define your own user interface for Header, which includes creating or deleting rules and groups and to customize the AND/OR condition and NOT condition options. To implement header template in querybuilder, you can create the user interface using x-template and assign the values when requestType is header-template-create in actionBegin event.

In the following sample dropdown, splitbutton and button are used as the custom components in the header.

INDEX.TS

```

import { QueryBuilder, RuleModel, ColumnsModel, ActionEventArgs } from
 '@syncfusion/ej2-querybuilder'
import { closest } from '@syncfusion/ej2-base';
import { DropDownButton, ItemModel, MenuEventArgs } from '@syncfusion/ej2-
splitbuttons';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { CheckBox } from '@syncfusion/ej2-buttons';
let filter: ColumnsModel [] = [
    { field: 'EmployeeID', label: 'EmployeeID', type: 'number'},
    { field: 'FirstName', label: 'FirstName', type: 'string'},
    { field: 'LastName', label: 'LastName', type: 'string'},
    { field: 'Age', label: 'Age', type: 'number'},
    { field: 'City', label: 'City', type: 'string'},
    { field: 'Country', label: 'Country', type: 'string'},
];
let importRules: RuleModel = {
    'condition': 'and', 'not': true,
    'rules': [{
        'label': 'Age',
        'field': 'Age',
        'type': 'number',
        'operator': 'equal',
        'value': 34
    },
    {
        'label': 'LastName',
        'field': 'LastName',
        'type': 'string',
        'operator': 'equal',
        'value': 'vinit'
    },
    {

```

```

        'condition': 'or',
        'rules': [{
            'label': 'Age',
            'field': 'Age',
            'type': 'number',
            'operator': 'equal',
            'value': 34
        }]
    }];
};
let queryBldrObj: QueryBuilder = new QueryBuilder({
    columns: filter,
    width: '100%',
    rule: importRules,
    headerTemplate: '#headerTemplate',
    actionBegin: actionBegin,
    enableNotCondition: true
});
queryBldrObj.appendTo('#querybuilder');
function actionBegin(args: ActionEventArgs): void {
    if (args.requestType === 'header-template-create') {
        let checkBoxObj: CheckBox = new CheckBox({
            label: 'NOT',
            checked: args.notCondition,
            change: function(e:any){
                queryBldrObj.notifyChange(e.checked,e.event.target,
'not')
            }
        });
        checkBoxObj.appendTo('#' + args.ruleID + '_notoption');
        let ds: { [key: string]: Object }[] = [{ 'key': 'AND', 'value':
'and'},{ 'key': 'OR', 'value': 'or'}];
        let btnObj: DropDownList= new DropDownList({
            dataSource: ds,
            fields: { text: 'key', value: 'value' },
            value: args.condition,
            cssClass: 'e-custom-group-btn e-active-toggle',
            change: (e: any) => {
                queryBldrObj.notifyChange(e.value, e.element,
'condition');
            }
        });
        btnObj.appendTo('#' + args.ruleID + '_cndtnbtn');
        let ddbitems: ItemModel[] = [
            { text: 'AddGroup', iconCss: 'e-icons e-add-icon e-addgroup'
},
            { text: 'AddCondition', iconCss: 'e-icons e-add-icon e-
addrule' }
        ];
        let addbtn: DropDownButton = new DropDownButton({
            items: ddbitems,
            cssClass: 'e-round e-small e-caret-hide e-addrulegroup',
            iconCss: 'e-icons e-add-icon',
            select: function(event: MenuEventArgs) {
                let addbtn: Element = closest(event.element, '.e-
dropdown-popup'); let ddb: string[] = addbtn.id.split('_');
                if (event.item.text === 'AddGroup') {

```



```

        queryBldrObj.addGroups([{condition: 'and', 'rules':
[{}], not: false}], ddb[1]);
        } else if (event.item.text === 'AddCondition') {
            queryBldrObj.addRules([{}], ddb[1]);
        }
    });
    addbtn.appendTo('#' + args.ruleID + '_addbtn');
    let deleteGroup: Element =
document.getElementById(args.ruleID).querySelector('.e-delete-btn');
    if (deleteGroup) {
        (deleteGroup as HTMLElement).onclick = function (e:any) {
            queryBldrObj.deleteGroup(closest(e.target.offsetParent,
'.e-group-container'));
        }
    }
}
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="querybuilder"></div>

```

```

</div>
<script id="headerTemplate" type="text/x-template">
  <div class = 'e-groupheader'>
    ${if(notCondition !== undefined)}
      <button class='e-cb-wrapper'>
        <input type="checkbox" class= "e-not"
id='${ruleID}_notoption' checked='${notCondition}'>
        </button>
      ${/if}
      <input type="text" class= "e-custom-group-btn"
id='${ruleID}_cndtnbtn'>
      <button id = '${ruleID}_addbtn' class = 'e-add-btn'></button>
      ${if(ruleID !== 'querybuilder_group0')}
        <button id= 'dltbtn' class = "e-btn e-delete-btn e-lib e-
small e-round e-icon-btn">
          <span class = 'e-btn-icon e-icons e-delete-icon'></span>
        </button>
      ${/if}
    </div>
</script>
<style type="text/css">
.e-query-builder .e-add-btn {
  margin-left: 10px;
  margin-right: 10px;
}
.e-query-builder .cndtnbtn.e-control.e-dropdownlist.e-lib.e-input {
  padding-left: 10px;
}
.e-query-builder span.e-custom-group-btn {
  max-width: 100px;
  border-radius: 5px !important;
  border-width: 1px !important;
}
.e-query-builder .e-custom-group-btn.e-input-focus::before, .e-custom-
group-btn.e-input-focus::after {
  background: transparent !important;
}
.e-query-builder .e-group-header .e-addrulegroup, .e-group-header .e-
delete-btn {
  border: 1px solid grey !important;
}
.e-query-builder .e-group-header .e-addrulegroup:hover, .e-group-header
.e-delete-btn:hover {
  border: 1px solid grey !important;
}
.e-query-builder .e-toggle{
  background: #317ab9;
  border-color: #317ab9;
  color: #fff;
}
.e-query-builder .e-cb-wrapper {
  margin-right: 5px;
  height: 32px;
  border-radius: 5px;
  border: 1px solid gray;
  background-color: white;
}

```

```

.e-query-builder .e-custom-group-btn{
    padding-left: 10px;
    height: 32px;
}
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column Template

Column Template allows you to define your own input widgets for columns. To implement [template](#), you can define the following functions

- **create**: Creates the custom component.
- **write**: Wire events for the custom component.
- **Destroy**: Destroy the custom component.

In the following sample, dropdown is used as the custom component in the PaymentMode column.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
import { getComponent } from '@syncfusion/ej2-base';
import { TextBox } from '@syncfusion/ej2-inputs';
import { DropDownList, MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
let expenseData: Object[] = [{
    'Category': 'Food',
    'PaymentMode': 'Credit Card',
    'TransactionType': 'Expense',
    'Description': 'Boiled peanuts',
    'Amount': '7',
    'Date': '06/01/2017'
}, {
    'Category': 'Food',
    'PaymentMode': 'Cash',
    'TransactionType': 'Expense',
    'Description': 'Peanuts in Coke',
    'Amount': '8',
    'Date': '06/04/2017'
}, {
    'Category': 'Food',
    'PaymentMode': 'Cash',
    'TransactionType': 'Expense',
    'Description': 'Palmetto Cheese, Mint julep',
    'Amount': '11',
    'Date': '06/04/2017'
}];

```

```

let elem: HTMLElement;
let inOperators: string [] = ['in', 'notin'];
let columnData: ColumnsModel[] = [
  { field: 'Category', label: 'Category', type: 'string' },
  { field: 'TransactionType', label: 'Transaction Type', type:
'boolean' },
  {
    field: 'PaymentMode', label: 'Payment Mode', type: 'string',
template: {
  create: () => {
    elem = document.createElement('input');
    elem.setAttribute('type', 'text');
    return elem;
  },
  destroy: (args: { elementId: string }) => {
    let multiSelect: MultiSelect =
(getComponent(document.getElementById(args.elementId), 'multiselect') as
MultiSelect);

    if (multiSelect) {
      multiSelect.destroy();
    }
    let dropdown: DropDownList =
(getComponent(document.getElementById(args.elementId), 'dropdownlist') as
DropDownList);

    if (dropdown) {
      dropdown.destroy();
    }
  },
  write: (args: { elements: Element, values: string[] |
string, operator: string }) => {
    let ds: string[] = ['Cash', 'Debit Card', 'Credit Card',
'Net Banking', 'Wallet'];
    if (inOperators.indexOf(args.operator) > -1) {
      let multiSelectObj: MultiSelect = new MultiSelect({
        dataSource: ds,
        value: args.values as string [],
        mode: 'CheckBox',
        placeholder: 'Select Transaction',
        change: (e: any) => {
          qryBldrObj.notifyChange(e.value, e.element);
        }
      });
      multiSelectObj.appendTo('#' + args.elements.id);
    } else {
      let dropDownObj: DropDownList = new DropDownList({
        dataSource: ds,
        value: args.values as string,
        change: (e: any) => {
          qryBldrObj.notifyChange(e.itemData.value,
e.element);
        }
      });
      dropDownObj.appendTo('#' + args.elements.id);
    }
  },
  operators: [

```

```

        { value: 'equal', key: 'Equal' },
        { value: 'notequal', key: 'Not Equal' },
        { value: 'in', key: 'In' },
        { value: 'notin', key: 'Not In' }
    ],
    },
    { field: 'Description', label: 'Description', type: 'string' },
    { field: 'Date', label: 'Date', type: 'date' },
    { field: 'Amount', label: 'Amount', type: 'number' }
];
let importRules: RuleModel = {
    'condition': 'and',
    'rules': [{
        'label': 'Payment Mode',
        'field': 'PaymentMode',
        'type': 'string',
        'operator': 'equal',
        'value': 'Cash'
    }
    ]
};
let qryBldrObj: QueryBuilder = new QueryBuilder({
    dataSource: expenseData,
    columns: columnData,
    width: '70%',
    rule: importRules
});
qryBldrObj.appendTo('#querybuilder');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div id="property"> </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using Template

Template allows you to define your own input widgets for columns. To implement template in querybuilder, you can create the user interface using **x-template** and assign the values through **actionBegin** event.

INDEX.TS

```

import { QueryBuilder, RuleModel, ColumnsModel, ActionEventArgs } from
'@syncfusion/ej2-querybuilder';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { CheckBox } from '@syncfusion/ej2-buttons';
let filter: ColumnsModel [] = [
    { field: "Category" label: "Category" type: "string" },
    { field: 'PaymentMode', label: 'PaymentMode', type: 'string', operators:
customOperators , template: "#paymentTemplate" },
    { field: 'TransactionType', label: 'TransactionType', type: 'string',
operators: customOperators, template: "#transactionTemplate" },
    { field: 'Description', label: 'Description', type: 'string' },
    { field: 'Date', label: 'Date', type: 'string' },
    { field: 'Amount', label: 'Amount', type: 'string' },
];
let customOperators: { [key: string]: Object }[] = [{value: 'equal', key:
'Equal'},
    {value: 'notequal', key: 'Not Equal'}];
let importRules: RuleModel = {
    'condition': 'and',
    'rules': [{
        'label': 'Transaction Type',
        'field': 'TransactionType',
        'type': 'string',
        'operator': 'equal',
        'value': 'Expense'
    },
    {

```

```

        'label': 'Payment Mode',
        'field': 'PaymentMode',
        'type': 'string',
        'operator': 'equal',
        'value': 'Cash'
    }]
    };
    let valueObj: Slider;
    let queryBldrObj: QueryBuilder = new QueryBuilder({
        columns: filter,
        rule: importRules,
        width: '100%',
        actionBegin: actionBegin
    });
    queryBldrObj.appendTo('#querybuilder');
    function actionBegin(args: ActionEventArgs): void {
        let ruleID: string = args.ruleID
        if (args.requestType === 'value-template-create') {
            let checkBoxObj: CheckBox = new CheckBox({
                label: 'Is Expense',
                checked: args.rule.value === "Expense" ? true: false,
                change: function(e:any){
                    let elem: HTMLElement =
document.getElementById(ruleID).querySelector('.e-rule-
value');qryBldrObj.notifyChange(e.checked === true ? 'Expense' : 'Income',
elem, 'value');
                }
            });
            checkBoxObj.appendTo('#' + args.ruleID + '_checkbox');
            let ds: string[] = ['Cash', 'Debit Card', 'Credit Card', 'Net
Banking'];
            let btnObj: DropDownList= new DropDownList({
                dataSource: ds,
                fields: { text: 'key', value: 'value' },
                value: args.rule.value,
                change: (e: any) => {
                    let elem: HTMLElement =
document.getElementById(ruleID).querySelector('.e-rule-
value');qryBldrObj.notifyChange(e.value as string, elem, 'value');
                }
            });
            btnObj.appendTo('#' + args.ruleID + '_paymentlist');
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Query Builder</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
    </div>
    <script id="paymentTemplate" type="text/x-template">
        <input type="text" class= "e-payment-list"
id='${ruleID}_paymentlist'>
    </script>
    <script id="transactionTemplate" type="text/x-template">
        <input type="checkbox" class= "e-checkbox" id='${ruleID}_checkbox'>
    </script>

    <style type="text/css">

    </style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Rule Template

Rule Template allows to define your own user interface for columns. To implement [ruleTemplate](#), you can create the user interface using [x-template](#) and assign the values through [actionBegin](#) event.

In the following sample, dropdown and slider are used as the custom component in the Age column and we have applied [greaterthanorequal](#) operator to this column.

INDEX.TS

```

import { QueryBuilder, RuleModel, ColumnsModel, ActionEventArgs } from
 '@syncfusion/ej2-querybuilder';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Slider } from '@syncfusion/ej2-inputs';
import { employeeData } from './datasource.ts';
import { DataManager, Query } from '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let filter: ColumnsModel [] = [
    { field: 'EmployeeID', label: 'EmployeeID', type: 'number'},
    { field: 'FirstName', label: 'FirstName', type: 'string'},
    { field: 'LastName', label: 'LastName', type: 'string'},
    { field: 'Age', label: 'Age', type: 'number', ruleTemplate:
'#ageTemplate'},
    { field: 'City', label: 'City', type: 'string'},
    { field: 'Country', label: 'Country', type: 'string'},
];
let importRules: RuleModel = {
    'condition': 'and',
    'rules': [{
        'label': 'Age',
        'field': 'Age',
        'type': 'number',
        'operator': 'greaterthanorequal',
        'value': 30
    }]
};
let fieldObj: DropDownList;
let valueObj: Slider;
let ruleID: string;
let queryBldrObj: QueryBuilder = new QueryBuilder({
    columns: filter,
    dataSource: employeeData,
    rule: importRules,
    width: '100%',
    actionBegin: actionBegin
});
queryBldrObj.appendTo('#querybuilder');
function actionBegin(args: ActionEventArgs): void {
    ruleID = args.ruleID;
    if (args.requestType === 'template-create') {
        args.rule.operator = 'greaterthanorequal';
        fieldObj = new DropDownList({
            dataSource: this.columns, // tslint:disable-line
            fields: args.fields,
            value: args.rule.field,
            change: (e: any) => {
                queryBldrObj.notifyChange(e.value, e.element, 'field');
            }
        });
        if (args.rule.value === '') {
            args.rule.value = 30;
        }
        valueObj = new Slider({
            value: args.rule.value as number,
            min: 30,

```

```

        max: 50,
        ticks: { placement: 'Before', largeStep: 5, smallStep: 1,
showSmallTicks: true },
        change: (e: any) => {
            let elem: HTMLElement = valueObj.element;
            queryBldrObj.notifyChange(e.value, elem, 'value');
            refreshTable(queryBldrObj.getRule(elem),
elem.id.split('_valuekey0')[0]);
        },
        created: () => {
            let elem: HTMLElement = valueObj.element;
            refreshTable(queryBldrObj.getRule(elem),
elem.id.split('_valuekey0')[0]);
        }
    });
    fieldObj.appendTo('#' + args.ruleID + '_filterkey');
    valueObj.appendTo('#' + args.ruleID + '_valuekey0');
}
}
function refreshTable(rule: RuleModel, ruleID: string) {
    let template: string =
'<tr><td>${EmployeeID}</td><td>${FirstName}</td><td>${Age}</td></tr>';
    let compiledFunction: Function = compile(template);
    let predicate = queryBldrObj.getPredicate({condition: 'and', rules:
[rule]});
    let dataManagerQuery: Query = new Query().select(['EmployeeID',
'FirstName', 'Age']).where(predicate);
    let result: Object[] = new
DataManager(employeeData).executeLocal(dataManagerQuery);
    let table: HTMLElement = (<HTMLElement>document.querySelector('#' +
ruleID + '_section #datatable'));
    if (result.length) {
        table.style.display = 'block';
    } else {
        table.style.display = 'none';
    }
    table.querySelector('tbody').innerHTML = '';
    result.forEach((data: Object) => {
table.querySelector('tbody').appendChild(compiledFunction(data)[0].querySele
ctor('tr'));
    });
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Query Builder</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-querybuilder/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
    </div>
    <script id="ageTemplate" type="text/x-template">
        <div class="e-rule e-rule-template">
            <div class="e-rule-filter e-custom-filter">
                <input id = ${ruleID}_filterkey class='e-filter-input'>
            </div>
            <div>
                <div class="e-value e-rule-value e-slider-
value">
                    <div id = ${ruleID}_valuekey0
class="ticks_slider"></div>
                </div>
                <div class="e-rule-btn">
                    <button id=${ruleID}_option
onclick="myFunction(${ruleID})" class="e-primary e-btn e-small">
                        View Details
                    </button>
                    <button class="e-remove-rule e-rule-
delete e-css e-btn e-small e-round">
                        <span class="e-btn-icon e-
icons e-delete-icon"/>
                    </button>
                </div>
            </div>
            <div id=${ruleID}_section class="e-rule-value-group e-hide">
                <div>
                    <table id='datatable' class='e-table e-rule-table'
style='display:none'>
                        <thead>
<tr><th>EmployeeID</th><th>FirstName</th><th>Age</th></tr>

```

```

        </thead>
        <tbody>
        </tbody>
    </table>
</div>
</div>
</div>
</script>
<script type="text/javascript">
    function myFunction(ruleElem) {
        var element = document.getElementById(ruleElem.id + '_section');
        if (element.className.indexOf('e-hide') > -1) {
            element.className = element.className.replace('e-hide', '');
            document.getElementById(ruleElem.id + '_option').textContent
= 'Hide Details';
        } else {
            element.className += ' e-hide';
            document.getElementById(ruleElem.id + '_option').textContent
= 'View Details';
        }
    }
</script>
<style type="text/css">
    .e-query-builder .e-custom-filter {
        width: 40% !important;
    }
    .e-query-builder .e-slider-value {
        width: 40% !important;
        padding: 12px 0 12px 0 !important;
        display: inline-block;
    }
    .e-query-builder .e-slider-container.e-horizontal {
        padding: 0 0 0 18px;
        height: 0;
    }
    .e-query-builder .e-slider-container.e-horizontal .e-slider
{
        top: calc(50% - 10px);
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Import export in EJ2 JavaScript Query builder control

Importing allows you to view or edit the predefined conditions which is available in JSON or SQL. You can import the conditions either initial rendering or post rendering.

Initial rendering

To apply conditions initially, you can define the [rule](#). Here, you can import structured JSON object by defining the [rule](#) property.

INDEX.TS

```
import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
let hardwareData: Object[] = [{
  'TaskID': 1,
  'Name': 'Lenovo Yoga',
  'Category': 'Laptop',
  'SerialNo': 'CB27932009',
  'InvoiceNo': 'INV-2878',
  'Status': 'Assigned'
},
{
  'TaskID': 2,
  'Name': 'Acer Aspire',
  'Category': 'Others',
  'SerialNo': 'CB35728290',
  'InvoiceNo': 'INV-3456',
  'Status': 'In-repair'
},
{
  'TaskID': 3,
  'Name': 'Apple MacBook',
  'Category': 'Laptop',
  'SerialNo': 'CB35628728',
  'InvoiceNo': 'INV-2763',
  'Status': 'In-repair'
}
];
let columnData: ColumnsModel[] = [
  { field: 'TaskID', label: 'Task ID', type: 'number' },
  { field: 'Name', label: 'Name', type: 'string' },
  { field: 'Category', label: 'Category', type: 'string' },
  { field: 'SerialNo', label: 'Serial No', type: 'string' },
  { field: 'InvoiceNo', label: 'Invoice No', type: 'string' },
  { field: 'Status', label: 'Status', type: 'string' }
];
let importRules: RuleModel = {
  'condition': 'or',
  'rules': [{
    'label': 'Category',
    'field': 'Category',
    'type': 'string',
    'operator': 'equal',
    'value': 'Laptop'
  },
  {
    'condition': 'and',
    'rules': [{
      'label': 'Status',
      'field': 'Status',
      'type': 'string',
      'operator': 'notequal',
      'value': 'Pending'
    }
  ]
}
```

```

        },
        {
            'label': 'Task ID',
            'field': 'TaskID',
            'type': 'number',
            'operator': 'equal',
            'value': 5675
        }
    ]
}

let gryBldrObj: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: hardwareData,
    columns: columnData,
    rule: importRules,
});
gryBldrObj.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="querybuilder"></div>

```

```

        <div id="property"> </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Post rendering

Importing from JSON

You can set the conditions from structured JSON object through the [setRules](#) method.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
import { Button } from '@syncfusion/ej2-buttons';
let hardwareData: Object[] = [{
    'TaskID': 1,
    'Name': 'Lenovo Yoga',
    'Category': 'Laptop',
    'SerialNo': 'CB27932009',
    'InvoiceNo': 'INV-2878',
    'Status': 'Assigned'
},
{
    'TaskID': 2,
    'Name': 'Acer Aspire',
    'Category': 'Others',
    'SerialNo': 'CB35728290',
    'InvoiceNo': 'INV-3456',
    'Status': 'In-repair'
},
{
    'TaskID': 3,
    'Name': 'Apple MacBook',
    'Category': 'Laptop',
    'SerialNo': 'CB35628728',
    'InvoiceNo': 'INV-2763',
    'Status': 'In-repair'
}
];
let columnData: ColumnsModel[] = [
    { field: 'TaskID', label: 'Task ID', type: 'number' },
    { field: 'Name', label: 'Name', type: 'string' },
    { field: 'Category', label: 'Category', type: 'string' },
    { field: 'SerialNo', label: 'Serial No', type: 'string' },
    { field: 'InvoiceNo', label: 'Invoice No', type: 'string' },
    { field: 'Status', label: 'Status', type: 'string' }
];
let importRules: RuleModel = {
    'condition': 'or',
    'rules': [{
        'label': 'Category',

```

```

        'field': 'Category',
        'type': 'string',
        'operator': 'equal',
        'value': 'Laptop'
    },
    {
        'condition': 'and',
        'rules': [{
            'label': 'Status',
            'field': 'Status',
            'type': 'string',
            'operator': 'notequal',
            'value': 'Pending'
        },
        {
            'label': 'Task ID',
            'field': 'TaskID',
            'type': 'number',
            'operator': 'equal',
            'value': 5675
        }
    ]
}
];
let qryBldrObj: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: hardwareData,
    columns: columnData
});
qryBldrObj.appendTo('#querybuilder');
let button = new Button({cssClass: `e-primary`, content: 'set
Rules'}, '#importjson');
document.getElementById('importjson').onclick = (): void => {
    qryBldrObj.setRules(importRules);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div class="e-qb-button">
            <button id="importjson" class="e-btn e-primary">Set
Rules</button>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Importing from SQL

You can set the conditions from SQL query through the [setRulesFromSql](#) method.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-
querybuilder';
import { Button } from '@syncfusion/ej2-buttons';
let hardwareData: Object[] = [{
    'TaskID': 1,
    'Name': 'Lenovo Yoga',
    'Category': 'Laptop',
    'SerialNo': 'CB27932009',
    'InvoiceNo': 'INV-2878',
    'Status': 'Assigned'
},
{
    'TaskID': 2,
    'Name': 'Acer Aspire',
    'Category': 'Others',
    'SerialNo': 'CB35728290',
    'InvoiceNo': 'INV-3456',
    'Status': 'In-repair'
},
];

```

```

{
    'TaskID': 3,
    'Name': 'Apple MacBook',
    'Category': 'Laptop',
    'SerialNo': 'CB35628728',
    'InvoiceNo': 'INV-2763',
    'Status': 'In-repair'
}];
let columnData: ColumnsModel[] = [
    { field: 'TaskID', label: 'Task ID', type: 'number' },
    { field: 'Name', label: 'Name', type: 'string' },
    { field: 'Category', label: 'Category', type: 'string' },
    { field: 'SerialNo', label: 'Serial No', type: 'string' },
    { field: 'InvoiceNo', label: 'Invoice No', type: 'string' },
    { field: 'Status', label: 'Status', type: 'string' }
];
let qryBldrObj: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: hardwareData,
    columns: columnData,
});
qryBldrObj.appendTo('#querybuilder');
let button: Button = new Button({cssClass: `e-primary`, content: 'set
Rules'}, '#importsql');
document.getElementById('importsql').onclick = (): void => {
    qryBldrObj.setRulesFromSql("TaskID = 1 and Status LIKE
('Assigned%')");
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Query Builder</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div class="e-qb-button">
            <button id="importsqli" class="e-btn e-primary">Set
Rules</button>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting

Exporting allows you to save or maintain the created conditions through the Query Builder. You can export the defined conditions by the following ways.

Exporting to JSON

You can export the defined conditions to structured JSON object through the [getRules](#) method.

Exporting to SQL

You can export the defined conditions to SQL query through the [getSqlFromRules](#) method.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-
querybuilder';
import { Button } from '@syncfusion/ej2-buttons';
import { Dialog } from '@syncfusion/ej2-popups';
let hardwareData: Object[] = [{
    'TaskID': 1,
    'Name': 'Lenovo Yoga',
    'Category': 'Laptop',
    'SerialNo': 'CB27932009',
    'InvoiceNo': 'INV-2878',
    'Status': 'Assigned'
},
{
    'TaskID': 2,
    'Name': 'Acer Aspire',
    'Category': 'Others',
    'SerialNo': 'CB35728290',
    'InvoiceNo': 'INV-3456',
    'Status': 'In-repair'
}
];

```

```

    },
    {
        'TaskID': 3,
        'Name': 'Apple MacBook',
        'Category': 'Laptop',
        'SerialNo': 'CB35628728',
        'InvoiceNo': 'INV-2763',
        'Status': 'In-repair'
    }
  ]];
  let columnData: ColumnsModel[] = [
    { field: 'TaskID', label: 'Task ID', type: 'number' },
    { field: 'Name', label: 'Name', type: 'string' },
    { field: 'Category', label: 'Category', type: 'string' },
    { field: 'SerialNo', label: 'Serial No', type: 'string' },
    { field: 'InvoiceNo', label: 'Invoice No', type: 'string' },
    { field: 'Status', label: 'Status', type: 'string' }
  ];
  let importRules: RuleModel = {
    'condition': 'or',
    'rules': [{
      'label': 'Category',
      'field': 'Category',
      'type': 'string',
      'operator': 'equal',
      'value': 'Laptop'
    }]
  };
  let qryBldrObj: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: hardwareData,
    columns: columnData,
    rule: importRules
  });
  qryBldrObj.appendTo('#querybuilder');
  let dialogObj: Dialog = new Dialog({
    header: 'Query builder',
    height: 'auto',
    animationSettings: { effect: 'Zoom', duration: 400 },
    showCloseIcon: true,
    width: '50%',
    visible: false
  });
  dialogObj.appendTo('#defaultdialog');
  let button: Button = new Button({cssClass: `e-primary`, content: 'get
sql', '#getsql'});
  document.getElementById('getsql').onclick = (): void => {
    dialogObj.content =
    qryBldrObj.getSqlFromRules(qryBldrObj.getRules());
    dialogObj.show();
  }
  button = new Button({cssClass: `e-primary`, content: 'get rule',
'#getrule'});
  document.getElementById('getrule').onclick = (): void => {
    let validRule: RuleModel =
    qryBldrObj.getValidRules(qryBldrObj.rule);
    dialogObj.content = '<pre>' + JSON.stringify(validRule, null, 4) +
'</pre>';

```

```

        dialogObj.show();
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Query Builder</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div class="e-qb-button">
            <button id="getsql" class="e-btn e-primary">get sql </button>
            <button id="getrule" class="e-btn e-primary"> get rule</button>
        </div>
        <div id="defaultdialog">
        </div>
    </div><script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Global local in EJ2 JavaScript Query builder control

The **Localization** library allows you to localize default text content of the Query Builder. The Query Builder has static text that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the **locale** value and translation object.

The following list of properties and its values are used in the Query Builder.

Locale key words	Text
-----	-----
AddGroup	Add Group
AddCondition	Add Condition
DeleteRule	Remove this condition
DeleteGroup	Delete group
Edit	EDIT
SelectField	Select a field
SelectOperator	Select operator
StartsWith	Starts With
EndsWith	Ends With
Contains	Contains
Equal	Equal
NotEqual	Not Equal
LessThan	Less Than
LessThanOrEqual	Less Than Or Equal
GreaterThan	Greater Than
GreaterThanOrEqual	Greater Than Or Equal
Between	Between
NotBetween	Not Between
In	In
NotIn	Not In
Remove	REMOVE
ValidationMessage	This field is required
SummaryViewTitle	Summary View
OtherFields	Other Fields
AND	AND
OR	OR
SelectValue	Enter Value

| IsEmpty | Is Empty |

| IsNotEmpty | Is Not Empty |

| IsNull | Is Null |

| IsNotNull | Is Not Null |

| True | True |

| False | False |

INDEX.TS

```
import { L10n } from '@syncfusion/ej2-base';
import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
let hardwareData: Object[] = [
    {
        'TaskID': 1,
        'Name': 'Lenovo Yoga',
        'Category': 'Laptop',
        'SerialNo': 'CB27932009',
        'InvoiceNo': 'INV-2878',
        'Status': 'Assigned'
    },
    {
        'TaskID': 2,
        'Name': 'Acer Aspire',
        'Category': 'Others',
        'SerialNo': 'CB35728290',
        'InvoiceNo': 'INV-3456',
        'Status': 'In-repair'
    },
    {
        'TaskID': 3,
        'Name': 'Apple MacBook',
        'Category': 'Laptop',
        'SerialNo': 'CB35628728',
        'InvoiceNo': 'INV-2763',
        'Status': 'In-repair'
    }
];
L10n.load({
    'de-DE': {
        'querybuilder': {
            'AddGroup': 'Gruppe hinzufügen',
            'AddCondition': 'Bedingung hinzufügen',
            'DeleteRule': 'Entfernen Sie diesen Zustand',
            'DeleteGroup': 'Gruppe löschen',
            'Edit': 'BEARBEITEN',
            'SelectField': 'Wählen Sie ein Feld aus',
            'DeleteRule': 'Entfernen Sie diesen Zustand',
            'DeleteGroup': 'Gruppe löschen',
            'SelectOperator': 'Operator auswählen',
            'StartsWith': 'Beginnt mit',
            'EndsWith': 'Endet mit',
            'Contains': 'Enthält',
            'Equal': 'Gleich',
            'NotEqual': 'Nicht gleich',
            'LessThan': 'Weniger als',
```

```

        'LessThanOrEqual': 'Weniger als oder gleich',
        'GreaterThan': 'Größer als',
        'GreaterThanOrEqual': 'Größer als oder gleich',
        'Between': 'Zwischen',
        'NotBetween': 'Nicht zwischen',
        'In': 'Im',
        'NotIn': 'Nicht in',
        'Remove': 'LÖSCHEN',
        'ValidationMessage': 'Dieses Feld wird benötigt',
        'True': 'Wahr',
        'False': 'Falsch',
    }
}
});

let columnData: ColumnsModel[] = [
    { field: 'TaskID', label: 'Task ID', type: 'number' },
    { field: 'Name', label: 'Name', type: 'string' },
    { field: 'Category', label: 'Category', type: 'string' },
    { field: 'SerialNo', label: 'Serial No', type: 'string' },
    { field: 'InvoiceNo', label: 'Invoice No', type: 'string' },
    { field: 'Status', label: 'Status', type: 'string' }
];

let importRules: RuleModel = {
    'condition': 'or',
    'rules': [{
        'label': 'Category',
        'field': 'Category',
        'type': 'string',
        'operator': 'equal',
        'value': 'Laptop'
    }]
};

let queryBuilder: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: hardwareData,
    locale: 'de-DE',
    columns: columnData,
    rule: importRules,
});

queryBuilder.appendTo('#querybuilder');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div id="property"> </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Style and appearance in EJ2 JavaScript Query builder control

To modify the QueryBuilder appearance, you need to override the default CSS of QueryBuilder component. Please find the list of CSS classes and its corresponding section in QueryBuilder component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class | Purpose of Class

|.e-group-header .e-btn|To customize the condition button in querybuilder

|.e-group-body .e-rule-container|To customize the querybuilder rule container

|.e-group-container .e-group-header .e-dropdown-btn|To customize the querybuilder Add group/condition button

|.e-query-builder .e-group-header .e-deletegroup|To customize the querybuilder Delete group button

|.e-query-builder .e-rule-field .e-rule-value-delete .e-rule-delete|To customize the querybuilder Delete condition button

|.e-query-builder .e-rule-list > ::after, .e-query-builder .e-rule-list > ::before|To customize the querybuilder group joining line

|.e-query-builder .e-rule-container.e-joined-rule|To customize the querybuilder condition joining line

Accessibility in EJ2 JavaScript Query builder control

The Query Builder component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Query Builder component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

WAI-ARIA (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components.

The following list of ARIA attributes is used in Query Builder.

Attributes	Purpose
---	---
role	Indicates the query builder component.

Keyboard interaction

The Query Builder component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Query Builder component.

Press	To do this
---	---
Tab / Shift + Tab	To focus the next item in the rule.

Ensuring accessibility

The Query Builder component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Query Builder component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Query Builder component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

How To

Display mode in EJ2 JavaScript Query builder control

Display options allows you to view the Query Builder in Vertically or Horizontally. For this, you should use the [displayMode](#) property.

INDEX.TS

```
import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
let employeeData: Object[] = [{
    'EmployeeID': 1,
    'FirstName': 'Nancy',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
    'City': 'Seattle',
    'Country': 'USA'
},
{
```

```

        'EmployeeID': 2,
        'FirstName': 'Andrew',
        'Title': 'Vice President',
        'TitleOfCourtesy': 'Dr.',
        'HireDate': '21/04/2003',
        'City': 'Tacoma',
        'Country': 'USA'
    },
    {
        'EmployeeID': 3,
        'FirstName': 'Janet',
        'Title': 'Sales Representative',
        'TitleOfCourtesy': 'Ms.',
        'HireDate': '22/07/2001',
        'City': 'Kirkland',
        'Country': 'USA'
    }
];
let columnData: ColumnsModel[] = [
    {
        field: 'EmployeeID', label: 'EmployeeID', type: 'number'
    },
    { field: 'FirstName', label: 'FirstName', type: 'string' },
    { field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type:
'boolean', values: ['Mr.', 'Mrs.'] },
    { field: 'Title', label: 'Title', type: 'string' },
    { field: 'HireDate', label: 'HireDate', type: 'date', format:
'dd/MM/yyyy' },
    { field: 'Country', label: 'Country', type: 'string' },
    { field: 'City', label: 'City', type: 'string' }
];
let importRules: RuleModel = {
    'condition': 'and',
    'rules': [{
        'label': 'EmployeeID',
        'field': 'EmployeeID',
        'type': 'number',
        'operator': 'equal',
        'value': 1001
    },
    {
        'label': 'Title',
        'field': 'Title',
        'type': 'string',
        'operator': 'equal',
        'value': 'Sales Manager'
    }
    ]
};
let qryBldrObj: QueryBuilder = new QueryBuilder({
    width: '50%',
    dataSource: employeeData,
    columns: columnData,
    rule: importRules,
    displayMode: "Vertical"
});
qryBldrObj.appendTo('#querybuilder');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="querybuilder"></div>
    <div id="property"> </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The default view in the desktop mode is Horizontal.

The default view in the mobile mode is Vertical.

Sort columns in EJ2 JavaScript Query builder control

SortDirection allows you to sort the columns bounded to the Query Builder to view the columns by ascending or descending order. You should set the [sortDirection](#) property to sort the fields.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
let employeeData: Object[] = [{
    'EmployeeID': 1,
    'FirstName': 'Nancy',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
    'City': 'Seattle',
    'Country': 'USA'
},
{
    'EmployeeID': 2,
    'FirstName': 'Andrew',
    'Title': 'Vice President',
    'TitleOfCourtesy': 'Dr.',
    'HireDate': '21/04/2003',
    'City': 'Tacoma',
    'Country': 'USA'
},
{
    'EmployeeID': 3,
    'FirstName': 'Janet',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
    'City': 'Kirkland',
    'Country': 'USA'
}
];
let columnData: ColumnsModel[] = [
    {
        field: 'EmployeeID', label: 'EmployeeID', type: 'number'
    },
    { field: 'FirstName', label: 'FirstName', type: 'string' },
    { field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type: 'boolean', values: ['Mr.', 'Mrs.'] },
    { field: 'Title', label: 'Title', type: 'string' },
    { field: 'HireDate', label: 'HireDate', type: 'date', format: 'dd/MM/yyyy' },
    { field: 'Country', label: 'Country', type: 'string' },
    { field: 'City', label: 'City', type: 'string' }
];
let importRules: RuleModel = {
    'condition': 'and',
    'rules': [{
        'label': 'EmployeeID',
        'field': 'EmployeeID',
        'type': 'number',
        'operator': 'equal',
        'value': 1001
    },
    {
        'label': 'Title',
        'field': 'Title',
        'type': 'string',

```

```

        'operator': 'equal',
        'value': 'Sales Manager'
    }]
};
let qryBldrObj: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: employeeData,
    columns: columnData,
    rule: importRules,
    sortDirection: "Ascending"
});
qryBldrObj.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="querybuilder"></div>
    <div id="property"> </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Restrict groups in EJ2 JavaScript Query builder control

You can restrict the condition set by defining the [maxGroupCount](#) property. By default, the value is 5. In the below demo, the [maxGroupCount](#) is set to 2 .

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
let employeeData: Object[] = [{
    'EmployeeID': 1,
    'FirstName': 'Nancy',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
    'City': 'Seattle',
    'Country': 'USA'
},
{
    'EmployeeID': 2,
    'FirstName': 'Andrew',
    'Title': 'Vice President',
    'TitleOfCourtesy': 'Dr.',
    'HireDate': '21/04/2003',
    'City': 'Tacoma',
    'Country': 'USA'
},
{
    'EmployeeID': 3,
    'FirstName': 'Janet',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
    'City': 'Kirkland',
    'Country': 'USA'
}
];
let columnData: ColumnsModel[] = [
    {
        field: 'EmployeeID', label: 'Employee ID', type: 'number'
    },
    {
        field: 'FirstName', label: 'FirstName', type: 'string' },
    {
        field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type: 'boolean', values: ['Mr.', 'Mrs.'] },
    {
        field: 'Title', label: 'Title', type: 'string' },
    {
        field: 'HireDate', label: 'HireDate', type: 'date', format: 'dd/MM/yyyy' },
    {
        field: 'Country', label: 'Country', type: 'string' },
    {
        field: 'City', label: 'City', type: 'string' }
];
let importRules: RuleModel = {
    'condition': 'and',

```



```

        'rules': [{
            'label': 'Employee ID',
            'field': 'EmployeeID',
            'type': 'number',
            'operator': 'equal',
            'value': 1001
        },
        {
            'label': 'Title',
            'field': 'Title',
            'type': 'string',
            'operator': 'equal',
            'value': 'Sales Manager'
        }
    ]
};
let qryBldrObj: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: employeeData,
    columns: columnData,
    rule: importRules,
    maxGroupCount: 2
});
qryBldrObj.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div id="property"> </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can use this property in the mobile mode to restrict the nested group creation.

State persistence in EJ2 JavaScript Query builder control

State persistence allows you to maintain the current state in the browser's `localStorage` even if the browser is refreshed or if you move to the next page within the browser. State persistence stores the Query Builder's `rule` object in the local storage when the [enablePersistence](#) is defined to true.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-
querybuilder';
let employeeData: Object[] = [{
    'EmployeeID': 1,
    'FirstName': 'Nancy',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
    'City': 'Seattle',
    'Country': 'USA'
},
{
    'EmployeeID': 2,
    'FirstName': 'Andrew',
    'Title': 'Vice President',
    'TitleOfCourtesy': 'Dr.',
    'HireDate': '21/04/2003',
    'City': 'Tacoma',
    'Country': 'USA'
},
{
    'EmployeeID': 3,
    'FirstName': 'Janet',
    'Title': 'Sales Representative',
    'TitleOfCourtesy': 'Ms.',
    'HireDate': '22/07/2001',
    'City': 'Kirkland',
    'Country': 'USA'
}
];

```

```

let columnData: ColumnsModel[] = [
    {
        field: 'EmployeeID', label: 'Employee ID', type: 'number'
    },
    { field: 'FirstName', label: 'FirstName', type: 'string' },
    { field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type:
'boolean', values: ['Mr.', 'Mrs.'] },
    { field: 'Title', label: 'Title', type: 'string' },
    { field: 'HireDate', label: 'HireDate', type: 'date', format:
'dd/MM/yyyy' },
    { field: 'Country', label: 'Country', type: 'string' },
    { field: 'City', label: 'City', type: 'string' }
];
let importRules: RuleModel = {
    'condition': 'and',
    'rules': [{
        'label': 'Employee ID',
        'field': 'EmployeeID',
        'type': 'number',
        'operator': 'equal',
        'value': 1001
    },
    {
        'label': 'Title',
        'field': 'Title',
        'type': 'string',
        'operator': 'equal',
        'value': 'Sales Manager'
    }
    ]
};
let qryBldrObj: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: employeeData,
    columns: columnData,
    rule: importRules,
    enablePersistence: true
});
qryBldrObj.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Query Builder</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="querybuilder"></div>
        <div id="property"> </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

RTL in EJ2 JavaScript Query builder control

RTL provides an option to switch the text direction and layout of the Query Builder component from right-to-left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL, set the [enableRtl](#) to true.

INDEX.TS

```

import { L10n } from '@syncfusion/ej2-base';
import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-
querybuilder';
let hardwareData: Object[] = [{
    'TaskID': 1,
    'Name': 'Lenovo Yoga',
    'Category': 'Laptop',
    'SerialNo': 'CB27932009',
    'InvoiceNo': 'INV-2878',
    'Status': 'Assigned'
},
{
    'TaskID': 2,
    'Name': 'Acer Aspire',
    'Category': 'Others',
    'SerialNo': 'CB35728290',

```

```

        'InvoiceNo': 'INV-3456',
        'Status': 'In-repair'
    },
    {
        'TaskID': 3,
        'Name': 'Apple MacBook',
        'Category': 'Laptop',
        'SerialNo': 'CB35628728',
        'InvoiceNo': 'INV-2763',
        'Status': 'In-repair'
    }
];
L10n.load({
    'ar-AE': {
        'querybuilder': {
            'AddGroup': 'إضافة مجموعة',
            'AddCondition': 'إضافة الشرط',
            'DeleteRule': 'أزل هذا الشرط',
            'DeleteGroup': 'حذف المجموعة',
            'Edit': 'تصحيح',
            'SelectField': 'اختر مجال',
            'DeleteRule': 'أزل هذا الشرط',
            'DeleteGroup': 'حذف المجموعة',
            'SelectOperator': 'حدد المشغل',
            'StartsWith': 'ابدا ب',
            'EndsWith': 'ينتهي مع',
            'Contains': 'يحتوي على',
            'Equal': 'مساو',
            'NotEqual': 'ليس متساوي',
            'LessThan': 'أقل من',
            'LessThanOrEqual': 'اصغر من أو يساوي',
            'GreaterThan': 'أكبر من',
            'GreaterThanOrEqual': 'أكبر من أو يساوي',
            'Between': 'ما بين',
            'NotBetween': 'ليس بينهما',
            'In': 'في',
            'NotIn': 'ليس في',
            'Remove': 'إزالة',
            'ValidationMessage': 'هذه الخانة مطلوبة'
        }
    }
});

let columnData: ColumnsModel[] = [
    { field: 'TaskID', label: 'Task ID', type: 'number' },
    { field: 'Name', label: 'Name', type: 'string' },
    { field: 'Category', label: 'Category', type: 'string' },
    { field: 'SerialNo', label: 'Serial No', type: 'string' },
    { field: 'InvoiceNo', label: 'Invoice No', type: 'string' },
    { field: 'Status', label: 'Status', type: 'string' }
];

let importRules: RuleModel = {
    'condition': 'or',
    'rules': [{
        'label': 'Category',
        'field': 'Category',
        'type': 'string',
        'operator': 'equal',
        'value': 'Laptop'
    }
]

```

```

    }}
  };
  let queryBuilder: QueryBuilder = new QueryBuilder({
    width: '70%',
    dataSource: hardwareData,
    columns: columnData,
    rule: importRules,
    enableRtl: true,
    locale: 'ar-AE',
  });
  queryBuilder.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="querybuilder"></div>
    <div id="property"> </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Summary view in EJ2 JavaScript Query builder control

Summary view allows you to show or hide the filtered query. By default, the value is false. You can enable this feature by setting the [summaryView](#) property to true.

INDEX.TS

```

import { QueryBuilder, ColumnsModel, RuleModel } from '@syncfusion/ej2-querybuilder';
import { employeeData } from './datasource.ts';
let columnData: ColumnsModel[] = [
    {
        field: 'EmployeeID', label: 'EmployeeID', type: 'number'
    },
    { field: 'FirstName', label: 'FirstName', type: 'string' },
    { field: 'TitleOfCourtesy', label: 'Title Of Courtesy', type: 'boolean', values: ['Mr.', 'Mrs.'] },
    { field: 'Title', label: 'Title', type: 'string' },
    { field: 'HireDate', label: 'HireDate', type: 'date', format: 'dd/MM/yyyy' },
    { field: 'Country', label: 'Country', type: 'string' },
    { field: 'City', label: 'City', type: 'string' }
];
let importRules: RuleModel = {
    'condition': 'and',
    'rules': [{
        'label': 'EmployeeID',
        'field': 'EmployeeID',
        'type': 'number',
        'operator': 'notequal',
        'value': '5'
    },
    {
        'condition': 'or',
        'rules': [{
            'label': 'Title Of Courtesy',
            'field': 'TitleOfCourtesy',
            'type': 'string',
            'operator': 'equal',
            'value': 'Mr.'
        },
        {
            'label': 'Country',
            'field': 'Country',
            'type': 'string',
            'operator': 'equal',
            'value': 'USA'
        }
    ],
    {
        'condition': 'and',
        'rules': [{
            'label': 'City',

```

```

        'field': 'City',
        'type': 'string',
        'operator': 'equal',
        'value': 'London'
    }]
  }]
  });
let qryBldrObj: QueryBuilder = new QueryBuilder({
  width: '30%',
  dataSource: employeeData,
  columns: columnData,
  rule: importRules,
  summaryView: "true"
});
qryBldrObj.appendTo('#querybuilder');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Query Builder</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
querybuilder/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="querybuilder"></div>
    <div id="property"> </div>

```



```
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```