Ruby - Feature #19071

Add Time#deconstruct, #deconstruct_keys, and #to_h

10/19/2022 07:21 PM - zverok (Victor Shepelev)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

I believe that Time being suitable for pattern-matching is a reasonable feature with many possible usages, which will increase usability of Time and would be a good show case for pattern matching.

Implementation decisions

Time#deconstruct:

- returns time components in order [year, month, mday, hour, min, sec, subsec]
- I believe the highest-to-lowest order is the only reasonable/guessable, and there is no point to put into the array *all* of the time information available (e.g. zone, wday, yday)
- I am not sure (and open to discussion) about subsec. It seems to me the most basic sub-second unit of Time, but I might be wrong; also, it might be not that useful for array deconstruction.

Possible usage:

```
case tm
in [...2022, *]
  puts "previous year"
in [2022, 1..6, *]
  puts "Q1-2"
in [2022, 7..9, *]
  puts "Q3"
in [2022, month, day, *]
  puts "Current quarter, #{day}/#{month}"
end
```

Time#deconstruct_keys:

- chosen keys: [:year, :month, :day, :yday, :wday, :hour, :min, :sec, :subsec, :dst, :zone]
- I am open to discussing whether we should include other subsecond units (or any whatsoever)
- It might be useful (but too loose interface) to support mon as a synonym for month?.. But might be confusing if somebody will unpack the **rest
- day, not mday, seems most reasonable

Possible usages:

```
case t
in year: ...2022
  puts "too old"
in month: ..9
  puts "quarter 1-3"
in wday: 1..5, month:
  puts "working day in month #{month}"
end

if t in Time(wday: 3, day: ..7)
  puts "first Wednesday of the month"
end
```

Time#to_h:

- added on a "why not" basis :) As we already have "convert to hash" in the form of deconstruct_keys(nil), having a canonic form seems harmles. Open for discussion.
- keys are the same as for deconstruct_keys

Pull request: https://github.com/ruby/ruby/pull/6594

History

#1 - 10/19/2022 07:47 PM - bdewater (Bart de Water)

My first reaction seeing the example usage: they seem more for Date than Time, but I can see this being useful for both :)

#2 - 10/20/2022 10:42 PM - matz (Yukihiro Matsumoto)

I agree with adding deconstruct_keys but not others (deconstruct and to_h).

- positional deconstruction force users to remember the meaning of order of elements (deconstruct)
- even though it's the same behavior to deconstruct_keys, there's no use-case for hash conversion from a time object (to_h)

Matz.

#3 - 10/21/2022 05:44 AM - sawa (Tsuyoshi Sawada)

I have another reason against deconstruct. The sequence [year, month, mday, hour, min, sec, subsec] is culturally biased and is arbitrarily selected. For example, following the fact that month is a highly irregular concept, some people avoid using it, and express the day with a combination of the week number and the day of week. Or, since the week number can be large, some people use the combination of the week-based quarter year, the week number within the quarter, and the day of the week.

#4 - 10/21/2022 08:39 AM - duerst (Martin Dürst)

sawa (Tsuyoshi Sawada) wrote in #note-3:

> I have another reason against deconstruct. The sequence [year, month, mday, hour, min, sec, subsec] is culturally biased and is arbitrarily selected.

There's a bit of cultural bias, but it's not an arbitrary selection, it's very clearly ordered from larger units to smaller, and this is a very widely used convention not only for time, but also for other units including currencies. Also, it's the order used in ISO 8601, the international standard for date and time formats. So very far from arbitrary.

> For example, following the fact that month is a highly irregular concept, some people avoid using it, and express the day with a combination of the week number and the day of week. Or, since the week number can be large, some people use the combination of the week-based quarter year, the week number within the quarter, and the day of the week.

There may be such people (I know that week number/weekday is very popular in Germany for business meetings), but overall this is a very small minority, and virtually everybody is familiar with (western) months.

#5 - 10/21/2022 09:16 AM - zverok (Victor Shepelev)

@matz (Yukihiro Matsumoto) Understood, thanks, I'll adjust the PR.

@sawa, for the sake of the argument

> The sequence [year, month, mday, hour, min, sec, subsec] is culturally biased and is arbitrarily selected.

I don't think this statement is fair. This sequence of units is not natural for some particular culture. E.g. 18:10 at Oct 21 would be written "10/21/22 6:10pm" in US, and "18:10 21.10.2022" in Ukraine, but "2022-10-21 18:10:00" is rather a common engineering implementation, logically structured for unambiguous parsing and sorting.

I understand there are different cultures out there (and for some, it is not "Oct 21, 2022" at all, but maybe "26 Tishrei, 5783" or "25 Rabi ul Awal 1444", or something else), but I believe it can be said with a high confidence that

1. when seeing a sequence of numbers starting with 2022, or year,, most of **developers** can guess the forthcoming units.
2. when trying to design or guess sequence of units of Time/Date representation *today* (with widespread standardization, JSON serializations etc.), the best guess would be close to ISO8601 Date & Time format.

#6 - 10/21/2022 06:44 PM - sawa (Tsuyoshi Sawada)

@duerst (Martin Dürst) (Martin Dürst), @zverok (Victor Shepelev) (Victor Shepelev)
In short, I think you two misinterpreted my comment. I never wrote that the order of the units is wrong. I am talking about the combination of the units (thus, I wrote *arbitrarily selected*). The proposal selected a particular subset of the units available in ISO 8601.

--

duerst (Martin Dürst) wrote in #note-4:

> it's very clearly ordered from larger units to smaller, and this is a very widely used convention not only for time, but also for other units including

> currencies. Also, it's the order used in ISO 8601, the international standard for date and time formats. So very far from arbitrary.

I am not sure why you are mentioning the order. If that combination [year, month, mday, hour, min, sec, subsec] were to be used, sure, ordering from larger units to smaller like that is the most logical way. No question. You mentioned currencies, but in Japanese, as you may know, that is the way it works for anything from family name-first name order, addresses, etc. In Japanese, even the AM/PM (□□/□□) is read after the day but before the hour because they (having half day span) are smaller units than a day but larger than hours.

zverok (Victor Shepelev) wrote in #note-5:

> I don't think this statement is fair. This sequence of units is not natural for some particular culture. E.g. 18:10 at Oct 21 would be written "10/21/22 6:10pm" in US, and "18:10 21.10.2022" in Ukraine, but "2022-10-21 18:10:00" is rather a common engineering implementation, logically structured for unambiguous parsing and sorting.

What is not fair? Of course "2022-10-21 18:10:00" is more logical and easier to parse than "18:10 at Oct 21", "10/21/22 6:10pm" or "18:10 21.10.2022". What is the point of bringing up somewhat convoluted formats and comparing with them?

duerst (Martin Dürst) wrote in #note-4:

> There may be such people (I know that week number/weekday is very popular in Germany for business meetings), but overall this is a very small minority, and virtually everybody is familiar with (western) months.

I did not say that people are not familiar with months. The concept is so wide spread that many people know it. The question is, whether they want to (be forced to) use it in their code.

For your information, the Quo Vadis weekly planners have gained a vast amount of users throughout the world. The page format using two facing pages allow for scheduling of a week, and they are numbered with the week number and the day of week. This is just one example, and many people throughout the world use similar planners/calendars as their choice.

zverok (Victor Shepelev) wrote in #note-5:

> I understand there are different cultures out there (and for some, it is not "Oct 21, 2022" at all, but maybe "26 Tishrei, 5783" or "25 Rabi ul Awal 1444", or something else), but I believe it can be said with a high confidence that
>
> 1. when seeing a sequence of numbers starting with 2022, or year,, most of **developers** can guess the forthcoming units.
> 2. when trying to design or guess sequence of units of Time/Date representation *today* (with widespread standardization, JSON serializations etc.), the best guess would be close to ISO8601 Date & Time format.

The very ISO8601 that you mention disallows commonplace descriptors of dates (or parts of dates) like "January", "Thursday", or "New Year's Day". So the formats you mention: "Oct 21, 2022", "26 Tishrei, 5783", or "25 Rabi ul Awal 1444" are all out of question.

And **the very ISO8601 that you mention does not force you to use calendar dates (month-based dates). It equally specifies week dates (week-based dates) and ordinal dates (dates based on the day of the year) as well.**

I do not understand the point of your statement:

> 1. when seeing a sequence of numbers starting with 2022, or year,, most of **developers** can guess the forthcoming units.

Sure, given the ISO 8601 format "2022-294", we can tell it is the 294th day of year 2022, or given the ISO 8601 format "2009-W53-7", we can tell that it is the Sunday of 53rd week of year 2009. What is wrong with that? And how is that related to my comment?

And actually, my comment is also related to the point Matz made against deconstruct:

> positional deconstruction force users to remember the meaning of order of elements

People would have to remember what comes next because it is not universal.

#7 - 10/22/2022 01:08 PM - zverok (Victor Shepelev)

@sawa With all due respect, I fail to see your point.

For all I know:

- most modern common serialization formats are using specifically "ISO8601 date and time" (which I specifically pointed at, NOT "everything ISO8601 is able to standardize") or something similar/compatible; even if they don't (RFC 2822, say), it is still based on day, month, year scheme, not week number or year day number (BTW, Ruby's Time#iso8601/.iso8601 don't support '2022-W42-6' and '2022-295' and I fail to remember any complaints about that in the last years)
- most of the time, Rubyists initiate new Time values with Time.new(year, month, day, ...) protocol, not with weeks or years. It is closest we have to time literal. So the idea of deconstruct was to be symmetrical to this, in the first place;

- as far as I can guess, the other languages do the same: in Python it would be datetime.datetime(2022, 10, 22, ...), in Java some LocalDate.of(2022, 10, 22), etc.

So, my point is:

1. There *are* many possible representations and ways of construction of time and datetime objects, and we all are well aware of it;
2. Colloquially, though, the "day, month, and year" is the most widely known to developers, and Y-m-d-H-M-S (save for timezones and punctuation) is the most expected engineering representation

In light of this, it would be hard for me to believe that there is any significant amount of engineers who would see in a generic code a sequence of numbers starting with year and would be stuck with guessing what the next number means.

Do you have real cases in mind of communities/codebases for which "year-week" or "year-yday" are *the first representation that comes to mind* while writing code? I am very curious to extend my understanding of the world if it is so.

Or was your point just to put me in my place, as it needs to be done when somebody proposes too much stuff? I am OK with this, too, but would prefer it to be spelled explicitly.

#8 - 11/22/2022 09:11 PM - zverok (Victor Shepelev)

- Status changed from Open to Closed

https://github.com/ruby/ruby/pull/6594 merged.