Ruby - Feature #13784

Add Enumerable#filter as an alias of Enumerable#select

08/04/2017 04:20 PM - davidarnold (David Arnold)

Status:	Closed
Priority:	Normal
Assignee:	Eregon (Benoit Daloze)
Target version:	2.6

Description

Ruby has a full set of functional tools in the Enumerable module under the "-ect" methods (viz. collect, select, inject). However the usual industry terms for these are map, filter, and reduce.

For example, Swift, Python, and ECMAScript all use the names map, filter, and reduce to describe these methods. Also, this language independent MIT course uses map, filter and reduce: http://web.mit.edu/6.005/www/fa15/classes/25-map-filter-reduce/

Ruby has aliases for map and reduce, but filter is noticeably absent. This feature request is simply to add an alias to Enumerable for filter. This will ease the transition of developers from other languages to Ruby.

Desired behavior:

[:foo, :bar].filter { |x| x == :foo } # => [:foo]

Current behavior:

[:foo, :bar].filter { |x| x == :foo } # NoMethodError: undefined method `filter'

Related issues:

Related to Ruby - Feature #5663: Combined map/select method	**Closed**

Associated revisions

Revision b1a8c64483b5ba5e4a391aa68234e7bde6355034 - 02/25/2018 01:52 PM - Eregon (Benoit Daloze)

Add a new #filter alias for #select

- In Enumerable, Enumerator::Lazy, Array, Hash and Set
  [Feature #13784] [ruby-core:82285]
- Share specs for the various #select#select! methods and
  reuse them for #filter/#filter!.
- Add corresponding filter tests for select tests.
- Update NEWS.

[Fix GH-1824]

From: Alexander Patrick adp90@case.edu

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@62575 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b1a8c644 - 02/25/2018 01:52 PM - Eregon (Benoit Daloze)

Add a new #filter alias for #select

- In Enumerable, Enumerator::Lazy, Array, Hash and Set
  [Feature #13784] [ruby-core:82285]
- Share specs for the various #select#select! methods and
  reuse them for #filter/#filter!.
- Add corresponding filter tests for select tests.
- Update NEWS.

[Fix GH-1824]

From: Alexander Patrick adp90@case.edu

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@62575 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 08/05/2017 03:18 AM - shyouhei (Shyouhei Urabe)

+1. I don't always agree with new aliases but for this one. I actually wanted Enumerable#filter several times before.

#2 - 08/05/2017 07:54 PM - adp90 (Alexander Patrick)

I made a pull request: https://github.com/ruby/ruby/pull/1672.

#3 - 08/06/2017 05:28 PM - shevegen (Robert A. Heiler)

> However the usual industry terms for these are map, filter, and reduce.

I do not know whether these are "industry terms" per se, but I think that the map/collect alias example was once explained in that matz wanted to make it easier for people who are using other languages, to use ruby "out of the box" in a way that is more similar to how they think (and thus write and design code). Of course they can also modify ruby as-is and add the aliases on their own, which would also include filter, but in many ways I think that I agree with you that filter is used in other languages too, and since it is not yet used by ruby by default, it could find a use there. So I have no objection.

There is however had only one smaller issue - I think in terms of .select and .reject already as filters. They just do the opposite of each other in their selections - one does a positive "give me the matches" and the other the "give me the non-matches", or "reject the matches", which can already be inverted via '!' for example.

So I am slightly +1 on the suggestion, but the docs should explain why .filter is used as alias for .select but not .reject if this is added.

I guess the ruby core team may discuss this - ultimately matz has to think about it if it fits or does not fit or is similar to the reason for map/collect addition many years ago (I do not even know which ruby version had it... when I started to use ruby, map/collect was always there and I always used only .map and never .collect; and I guess there are people who are doing this in the other way too. More than one way to do something). :)

#4 - 08/06/2017 10:23 PM - davidarnold (David Arnold)

> > However the usual industry terms for these are map, filter, and reduce.
>
> I do not know whether these are "industry terms" per se [..]

I suppose there is not a truly objective measure of when something becomes an industry term, but here are 12 more languages that all refer to this function as "filter":

- Closure
- D
- Erlang
- Haskell
- F#
- OCaml
- Standard ML
- Prolog
- Java 8 (streams API)
- PHP (array_* functions)
- R
- Scala

Along with the first three I mentioned (Swift, Python, and ECMAScript), this seems to be somewhat of a preponderance in the industry, no?

> So I am slightly +1 on the suggestion, but the docs should explain why
> .filter is used as alias for .select but not .reject if this is added.

For a rationale, I would say that there are languages which do not call this function "filter", but I believe there are no languages with a function named filter that rejects based on the predicate.

However, I'm not sure why there would be an explanation of this method's name when there are no justifications of any of the other Enumerable names in the docs.

#5 - 08/08/2017 03:01 PM - mikegee (Michael Gee)

I prefer not adding this alias. I understand that it is beneficial for people familiar with other languages that have a "filter" function like this, but I believe "filter" is confusing for people without that familiarity.

The word "filter" implies a separation, but does not convey which part we are "keeping" like "select" and "reject" do. When I mentioned this ticket to my coworker, his initial reaction was that "filter" would be an alias for "reject".

#6 - 08/08/2017 03:31 PM - davidarnold (David Arnold)

> I understand that it is beneficial for people familiar with other languages that have a "filter" function like this, but I believe "filter" is confusing for people without that familiarity.

The great thing is that with aliases, you can choose to use whichever one you like better. If select, reject, or find_all makes the most sense, then go ahead and keep using it.

Why not add something that is greatly beneficial for people with experience in any of those 15 other languages? The documentation would make it clear what it does.

> When I mentioned this ticket to my coworker, his initial reaction was that "filter" would be an alias for "reject".

I suppose depending on your background, anything could be confusing. For example I would have guessed that select was map (since that exactly what it means in SQL and LINQ) and that collect would have been reduce (since you are going through the list and collecting them into a single value), and inject, well... I wouldn't have had any clue what that meant without reading the docs.

Which goes back to the point about documentation. Who (aside from Smalltalk developers) would have known what collect, select, and inject meant without reading the docs? But we read it, said "ok", and used them correctly.

If the documentation for filter states that it "returns an array containing all elements of enum for which the given block returns a true value" then that is what it does.

#7 - 08/08/2017 04:22 PM - zornme (Matt Zorn)

+1 to "filter". Martin Fowler refers to this method as "filter" in his articles about collection pipelines (https://martinfowler.com/articles/collection-pipeline/) and his post about the operation (https://martinfowler.com/articles/collection-pipeline/filter.html) speaks to the advantages and disadvantages to the term "select".

#8 - 08/08/2017 09:34 PM - adp90 (Alexander Patrick)

A potential concern could be language bloat. Having too many aliases for the same methods could be confusing.
I was curious which underlying methods have the most aliases, so I wrote a quick script to look through the code.

[C method name, number of aliases]

["lazy_super", 5]
["rb_hash_has_key", 4]
["proc_call", 4]
["wmap_has_key", 3]
["time_utc_offset", 3]

All other methods have 1 or 2 aliases. Furthermore, lazy_super (chunk and slice methods) and proc_call aliases each do different things so they don't really count.

Filter would be enum_find_all's third alias. That could be fine, but I'm interested in others' thoughts on this.

On another note, there is a small difference between 'find_all' and 'select', due to 'select' being overridden for hashes. (
https://stackoverflow.com/questions/20999192/is-find-all-and-select-the-same-thing)
It might be inconsequential, but the 'filter' method in my pull request acts more like 'find_all'.

hash = {a: 1, b: 2, c: 3, d: 4, e: 5, f: 6}

hash.select { |k,v| v.even? }
=> {:b=>2, :d=>4, :f=>6}

hash.find_all { |k,v| v.even? }
=> [[:b, 2], [:d, 4], [:f, 6]]

hash.filter { |k,v| v.even? }

=> [[:b, 2], [:d, 4], [:f, 6]]

#9 - 08/09/2017 03:25 AM - davidarnold (David Arnold)

> On another note, there is a small difference between 'find_all' and 'select', due to 'select' being overridden for hashes. (https://stackoverflow.com/questions/20999192/is-find-all-and-select-the-same-thing)
> It might be inconsequential, but the 'filter' method in my pull request acts more like 'find_all'.

This is a very interesting find, I was very surprised that select and find_all currently do not work the same way in Hash. I did some archeology on the commits and I believe this is an unintentional oversight.

Enumerable#select was added in 1999 for Ruby 1.4.0 as an alias for Enumerable#find_all, which already existed. Hash was an Enumerable, so it would have gotten these methods too.

Then in 2001, between Ruby 1.6.x and 1.8.0, Hash#select was overridden presumably to support a hash.select(key1, key2, ...) syntax for returning multiple values. If a block was passed, it still worked like Hash#find_all. Interestingly enough, this feature was deprecated in 2003 before 1.8's release.

Later in 2003, the deprecated non-block code path in Hash#select was removed for the release of 1.8.2. After this change, Hash#select would have worked like Enumerable#select again, making the override appears superfluous to me.

Much later in 2007, Hash#select was changed to its present form, returning a hash instead of an array, leaving Hash#find_all with the Enumerable implementation that still returns an array. This change was included for the release of Ruby 1.9.

So my guess is that only Hash#select got the new behavior since it already existed in hash.c whereas find_all was only defined in Enumerable. Assuming this feature is approved, I will open a separate bug to start a discussion about whether the discrepancy is intentional or if Hash#find_all should be changed to match Hash#select.

For the purpose of this feature request, I would leave the filter == find_all behavior the same. If the decision in the bug report is that Hash#find_all should match Hash#select, then an alias can be added for both Hash#find_all and Hash#filter.

#10 - 08/09/2017 04:52 AM - duerst (Martin Dürst)

I think adding filter as an alias of select is a good idea, because indeed many languages use that name.

As for confusability with reject, everybody who already has seen filter in another language will assume it's an alias for select, not for reject. Those who haven't will very quickly learn that when they use it the first time.

davidarnold (David Arnold) wrote:

> On another note, there is a small difference between 'find_all' and 'select', due to 'select' being overridden for hashes. (https://stackoverflow.com/questions/20999192/is-find-all-and-select-the-same-thing)
> It might be inconsequential, but the 'filter' method in my pull request acts more like 'find_all'.

> So my guess is that only Hash#select got the new behavior since it already existed in hash.c whereas find_all was only defined in Enumerable. Assuming this feature is approved, I will open a separate bug to start a discussion about whether the discrepancy is intentional or if Hash#find_all should be changed to match Hash#select.
>
> For the purpose of this feature request, I would leave the filter == find_all behavior the same. If the decision in the bug report is that Hash#find_all should match Hash#select, then an alias can be added for both Hash#find_all and Hash#filter.

No, please fix your bug. Your proposal and its title are explicitly to make filter an alias of select. Please don't make that dependent on another, separate 'bug'. It would be really inconvenient if we had to say "filter is an alias of select, except for Hash, where it's an alias of find_all.

#11 - 08/09/2017 06:51 AM - adp90 (Alexander Patrick)

davidarnold (David Arnold) wrote:

> For the purpose of this feature request, I would leave the filter == find_all behavior the same. If the decision in the bug report is that Hash#find_all should match Hash#select, then an alias can be added for both Hash#find_all and Hash#filter.

I decided to update the pull request so that 'filter' and 'filter!' would work like 'select' and 'select!' for Array, Hash, Set, SortedSet, and ENV. I added spec and tests for filter to each of these, and given find_all's strange lack of testing or documentation, basing it on select seemed like the best course of action.

'Find_all' only functions differently than 'select' for Hashes, but has no spec or tests outside of Enumerable, while select is well-documented and tested throughout Ruby. Given that it's an alias for select, it's also curious that there is no 'find_all!'

#12 - 08/09/2017 02:49 PM - davidarnold (David Arnold)

> No, please fix your bug. Your proposal and its title are explicitly to make filter an alias of select.

Not sure why this would be "my" bug to fix, I didn't have anything to do with the code that was written 10 years ago :) Also, my proposal is explicitly to make filter an alias of select *on Enumerable* -- check the title.

I do agree that the inconsistency is bad in Hash and would like to see it solved too, but I was a little worried that reaching out and changing Hash behavior would increase the risk that this feature request would be rejected. I think it's really important to get filter into Enumerable and wouldn't want to see it quashed by concerns around changing Hash.

#13 - 08/09/2017 02:52 PM - davidarnold (David Arnold)

> I decided to update the pull request so that 'filter' and 'filter!' would work like 'select' and 'select!' for Array, Hash, Set, SortedSet, and ENV. I added spec and tests for filter to each of these, and given find_all's strange lack of testing or documentation, basing it on select seemed like the best course of action.

Awesome! Thank you. I am beginning to get a little suspicious that find_all keeps getting left behind because almost no one uses that name. I personally haven't seen any books or online examples where someone favored find_all over select.

#14 - 08/31/2017 06:37 AM - shugo (Shugo Maeda)

+1 because I like the name filter_map for #5663.

#15 - 08/31/2017 06:38 AM - shugo (Shugo Maeda)

- Related to Feature #5663: Combined map/select method added

#16 - 08/31/2017 09:09 AM - matz (Yukihiro Matsumoto)

Sounds OK. One concern left is Hash#filter.

Matz.

#17 - 08/31/2017 01:02 PM - davidarnold (David Arnold)

matz (Yukihiro Matsumoto) wrote:

> Sounds OK. One concern left is Hash#filter.
>
> Matz.

I have added a bug to discuss the discrepancy in Hash's behavior: https://bugs.ruby-lang.org/issues/13795

#18 - 02/25/2018 01:52 PM - Eregon (Benoit Daloze)

- Status changed from Open to Closed

Applied in changeset trunk|r62575.

Add a new #filter alias for #select

- In Enumerable, Enumerator::Lazy, Array, Hash and Set
  [Feature #13784] [ruby-core:82285]
- Share specs for the various #select#select! methods and
  reuse them for #filter/#filter!.
- Add corresponding filter tests for select tests.
- Update NEWS.

[Fix GH-1824]

From: Alexander Patrick adp90@case.edu

#19 - 02/25/2018 02:02 PM - Eregon (Benoit Daloze)

- Assignee set to Eregon (Benoit Daloze)

- Target version set to 2.6

#20 - 02/27/2018 12:15 AM - shevegen (Robert A. Heiler)

I think this is a good change; I just noticed it from the NEWS
file at:

https://github.com/ruby/ruby/blob/trunk/NEWS

I think of .select and .reject as filters already - we filter
either for what we want to keep, or for what we want to discard.

At the least this is how my brain "remembers" this.

I tend to prefer .select, mostly because also how my brain works -
I find it easier to think in a "positive" way, e. g. which is why
I prefer to write code that uses "if condition" rather than
"unless condition" - the second variant takes me a bit longer to
process, if I actively think about it.

Both .select and .reject act as filters ultimately, but since I
myself try to write code in a way to prefer .select, it suits me
to see .filter being an alias to .select rather than an alias to
.reject.

I used to write code like this:

```
Dir['**/**'].reject {|entry| File.directory?(entry) }
```

Like to get mostly files; but I think it works just fine via
.select too, and using a test via File.file? (and perhaps also
testing for symlinks).

Sorry for the long addition here - I only just noticed that change
just now when Benoit made the change recently. :)