

Ruby - Feature #15323

[PATCH] Proposal: Add Enumerable#filter_map

11/20/2018 11:59 AM - alfonsojimenez (Alfonso Jiménez)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		
Description This is a proposal for a combined filter + map method (https://bugs.ruby-lang.org/issues/5663). This method both filters and maps the elements of an enumerable in just one iteration: <pre>(1..10).filter_map { i i * 2 if i.even? } #=> [4, 8, 12, 16, 20]</pre> GitHub PR: https://github.com/ruby/ruby/pull/2017		
Related issues: Related to Ruby - Feature #5663: Combined map/select method Closed		

Associated revisions

Revision 0acbddd1ed0d2302743525a5188cc5a0d6251680c - 05/23/2019 05:39 AM - alfonsojimenez (Alfonso Jiménez)

Adding Enumerable#filter_map

[Feature #15323]

Closes: <https://github.com/ruby/ruby/pull/2017>

Revision 0acbddd1ed0d2302743525a5188cc5a0d6251680c - 05/23/2019 05:39 AM - alfonsojimenez (Alfonso Jiménez)

Adding Enumerable#filter_map

[Feature #15323]

Closes: <https://github.com/ruby/ruby/pull/2017>

Revision 0acbddd1e - 05/23/2019 05:39 AM - alfonsojimenez (Alfonso Jiménez)

Adding Enumerable#filter_map

[Feature #15323]

Closes: <https://github.com/ruby/ruby/pull/2017>

History

#1 - 11/20/2018 12:01 PM - alfonsojimenez (Alfonso Jiménez)

- Description updated

#2 - 11/20/2018 02:23 PM - alfonsojimenez (Alfonso Jiménez)

- File deleted (0001-Adding-Enumerable-filter_map.patch)

#3 - 11/20/2018 02:24 PM - alfonsojimenez (Alfonso Jiménez)

- File 0001-Adding-Enumerable-filter_map.patch added

#4 - 11/20/2018 02:29 PM - tny (Tony Sunny)

Could't we use reduce for this?

```
(1..10).reduce([]) { |a, i| i.even? ? a << (i * 2) : a }
```

#5 - 11/20/2018 05:07 PM - shevegen (Robert A. Heiler)

I think the functionality, that is to combine .filter (be it select or reject, is secondary to me), and .map, could be useful. I don't really need it myself but I find it is not entirely out of the question that others may find it useful.

There is, IMO, only one real drawback, if we ignore the functionality aspect (where you'd have to ask matz anyway), and this is that I think the two-word methods can be quite clumsy.

Not just `.filter_map` but also `.yield_self`, which eventually had an alias called `.then`. If we ignore the question as to whether `.then` is a good name (or `.yield_self`), one advantage that `.then` has is that it is shorter.

Succinct expression is not always necessarily the best; but in this case, I think single-word methods are very often better than two-word methods.

`.reduce()` has, in my opinion, a slight other disadvantage, and that is that people have to explicitly pass an `[]` (as in the example here), which is not always easy to remember. (For me it is hard to remember because I rarely use `.reduce` either).

This is just my opinion, though. I do not really have any strong pro or con way about the feature itself; only a very tiny dislike of `.filter_map` as name. But it is not really a strong contra opinion either way. (My biggest look ahead is on ruby's `jit/mjit` ... :D)

#6 - 12/14/2018 07:58 PM - devpolish (Nardo Nykolyszyn)

```
(1..10).map { |e| e.even? ? (e * 2) : e }
```

#7 - 12/14/2018 08:32 PM - Anonymous

nardonykolyszyn@gmail.com wrote:

```
(1..10).map { |e| e.even? ? (e * 2) : e }
```

Yeah, but without `#filter` this is still an array with 10 elements.

#8 - 12/17/2018 03:43 AM - phluid61 (Matthew Kerwin)

tny (Tony Sunny) wrote:

Could't we use `reduce` for this?

```
(1..10).reduce([]) { |a, i| i.even? ? a << (i * 2) : a }
```

Yep, that's mentioned in the original ticket too. There's also `#each_with_object` that lets you write the block almost the same as in the proposal:

```
(1..10).each_with_object([]) { |i, a| a << i * 2 if i.even? }
```

The big difference here is you can capture `nil/false` values, because the filter test is explicitly separated from the map operation.

#9 - 02/13/2019 02:35 AM - shugo (Shugo Maeda)

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

#10 - 02/13/2019 02:41 AM - shugo (Shugo Maeda)

+1 for `filter_map`.

Matz agreed the feature itself before: <https://bugs.ruby-lang.org/issues/5663#note-42>

The name `filter_map` is good because other languages have similar names (e.g., `filter-map` in Scheme).

#11 - 04/25/2019 09:48 AM - alfonsojimenez (Alfonso Jiménez)

- File deleted (0001-Adding-Enumerable-filter_map.patch)

#12 - 04/25/2019 09:49 AM - alfonsojimenez (Alfonso Jiménez)

- File 0001-Adding-Enumerable-filter_map.patch added

#13 - 04/25/2019 09:54 AM - alfonsojimenez (Alfonso Jiménez)

I've updated the patch file increasing the ruby version in `spec/ruby/core/enumerable/filter_map_spec.rb`

`Enumerable#filter_map` was already accepted in the last developers meeting:

<https://docs.google.com/document/u/2/d/e/2PACX-1vTUCmj7aUdnMAduG0AZo0AdWK-9jvfXcB7DWYmzGtmPc0luIPGn7eLARoR5tBd6XUUB08W-hH74k-T/pub>

#14 - 04/30/2019 09:35 PM - greggzst (Grzegorz Jakubiak)

alfonsojimenez (Alfonso Jiménez) wrote:

I've updated the patch file increasing the ruby version in *spec/ruby/core/enumerable/filter_map_spec.rb*

Enumerable#filter_map was already accepted in the last developers meeting:

<https://docs.google.com/document/u/2/d/e/2PACX-1vTUCmj7aUdnMAduG0AZo0AdWK-9jvfXcB7DWYmzGtmPc0luIPGn7eLARoR5tBd6XUUB08W-hH74k-T/pub>

Does the syntax allow for this kind of code?

```
(1..10).filter_map(&:even?) { |i| i * 2 }
```

#15 - 05/22/2019 06:06 AM - matz (Yukihiro Matsumoto)

I accepted the proposal at the last developer meeting but forgot to post here.

I do reject having both block and block argument at the same time. [\[ruby-core:92505\]](#)

Regarding filter_map!, submit a new proposal, if you really needed (with the use-case).

Matz.

#16 - 05/23/2019 05:51 AM - alfonsojimenez (Alfonso Jiménez)

- Status changed from Open to Closed

Applied in changeset [git|0acbdd1ed0d2302743525a5188cc5a0d6251680c](https://github.com/ruby/ruby/pull/2530).

Adding Enumerable#filter_map

[Feature [#15323](#)]

Closes: <https://github.com/ruby/ruby/pull/2017>

#17 - 05/23/2019 03:46 PM - nobu (Nobuyoshi Nakada)

IIRC, at the last meeting (20190522), the conclusion was that this method should select non-nil values only, like as Array#compact.

Am I correct?

#18 - 10/06/2019 07:16 PM - jonathanhefner (Jonathan Hefner)

nobu (Nobuyoshi Nakada) wrote:

IIRC, at the last meeting (20190522), the conclusion was that this method should select non-nil values only, like as Array#compact.

Am I correct?

Checking master, it looks like this was not addressed. I agree it would be more intuitive for filter_map to behave like map{ ... }.compact. Rejecting false values seems like a "gotcha" / footgun.

I have submitted a PR: <https://github.com/ruby/ruby/pull/2530>

#19 - 10/06/2019 07:22 PM - Eregon (Benoit Daloze)

Isn't enum.filter_map { |e| ... } supposed to be (according to the name) the same as enum.map { |e| ... }.filter { |e| e }?

I'm not completely sure what is better, but to me it sounds surprising that a method with filter in its name filters differently than Enumerable#filter (which removes both false and nil values).

#20 - 10/06/2019 08:43 PM - jonathanhefner (Jonathan Hefner)

Eregon (Benoit Daloze) wrote:

but to me it sounds surprising that a method with filter in its name filters differently than Enumerable#filter (which removes both false and nil values).

Yes, I suppose that is surprising too... But, I think that throwing away false values is still a footgun. If I write:

```
records.filter_map{|record| record.send(field) if record.valid? }
```

I would expect to get a value for every valid record, no matter what field is.

Should the name of `filter_map` be reconsidered?

#21 - 10/08/2019 09:35 PM - Eregon (Benoit Daloze)

jonathanhefner (Jonathan Hefner) wrote:

Eregon (Benoit Daloze) wrote:

I would expect to get a value for every valid record, no matter what field is.

What if `record.send(field)` returns `nil`?

Then the only way is:

```
records.filter { |record| record.valid? }.map { |record| record.send(field) }
```

So this kind of issue is intrinsically there for `filter_map`.
`filter_map` is a footgun if `nil` can be returned.

I see the point that `nil` is like "missing element, filter it out" versus `false` being a regular value (and `true` too).

I tend to agree with you now, I think just filtering out `nil` values would be better than also removing `false`.

#22 - 10/11/2019 02:34 AM - inopinatus (Joshua GOODALL)

Could this take argument as well? I'd be keen for something in the spirit of `Enumerable#grep` but where an (optional) pattern evaluates the result of the block, rather than each element of the array. If omitted, there's the default test.

I'd love to be able to write `bottles.filter_map(18.., &:age).tally` or `dependencies.filter_map(/GPL/) { |lib| lib.license.name.upcase }`.

#23 - 11/04/2019 09:32 PM - shevegen (Robert A. Heiler)

Jonathan recently added this to the upcoming developer discussion (at <https://bugs.ruby-lang.org/issues/16262>), in regards as to whether the current behaviour is correct, or whether it should include non-`nil` values as well.

The question is whether the behaviour should be like `Array#compact` or not.

I can not say much about `Array#compact`, but from the name `filter_map` alone, I would assume that first a filter is used (as a "positive" `.select`), and then the `.map` is applied. If this reasoning makes any sense to anyone else, then I believe that the behaviour shown by `.filter_map` as-is is correct and should be retained. But this is just my opinion - my reasoning comes primarily from the name itself (`.filter_map` that is).

#24 - 11/04/2019 09:35 PM - shevegen (Robert A. Heiler)

Actually, after rereading what Jonathan wrote, he referred not to "true" values per se, but as to whether "non nil values" are to be included. So perhaps I misunderstood his comment. I think that filter still applies to the `.select` and should return what matches to the given criterium, so from this point of view I understand Jonathan's confusion. Either way I think it is best to define this clearly.

#25 - 11/28/2019 04:19 AM - nobu (Nobuyoshi Nakada)

It seems OK as the original proposal is same as the current behavior.

Files

0001-Adding-Enumerable-filter_map.patch	4.61 KB	04/25/2019	alfonsojimenez (Alfonso Jiménez)
---	---------	------------	----------------------------------