Ruby master - Feature #15145

chained mappings proposal

09/21/2018 07:27 PM - koenhandekyn (koen handekyn)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

propsal, have map accept array of method references to make chained mapping simpler with suggestion to implement it to behave like the &. operator so that intermediate nil values just ripple through as nil values

proposal allow

```
collection.map(&:instrument, &:issuer, &:name)
```

implementation is trivial

this as a conceptual alternative to (with important remark is that if first or second mapping returns nil values the above code will break, forcing a much more verbose notation)

```
collection.map(&:instrument).map(&:issuer).map(&:name)
```

proposal to be functional equivalent to

```
collection.map { |e| e&.instrument }.map { |e| e&.issuer }.map { |e| e&.name }
```

History

#1 - 09/21/2018 07:28 PM - koenhandekyn (koen handekyn)

- Description updated

#2 - 09/21/2018 10:51 PM - shevegen (Robert A. Heiler)

(I think you filed this in the wrong section; right now it is under Bugs, but it looks
like a feature so it should go into
https://bugs.ruby-lang.org/projects/ruby-trunk/issues?set_filter=1&tracker_id=2)

At any rate, to the suggestion itself:

I am already biased because I am not a big fan of object&.method for various reasons,
some of which are a personal preference (but also because my eyesight is not the best).

However had, you don't have to convince me - you ultimately only have to convince
matz (and it helps to convince the core team too).

Aside from any personal bias, I think there may be more objective reasons against the
proposal from a syntax and "meaning" point of view.

I may be mistaken, but I believe

```
collection.map(&:instrument)
```

always meant a to_proc call (e. g. https://stackoverflow.com/a/8793693/722915).

This was probably also one reason why it could not too easily be extended to
also allow for arguments passed into it, e. g. anything where you
pass stuff into the method that is called there.

Having object.map(&:foo) suddenly mean object.map { |e| e&.foo } is, I think,
re-defining existing behaviour. Perhaps there are cases where this new behaviour
would be undesired? But I may be wrong; I make use of & only very sparingly
so in the ruby code I write/use.

Aside from this, the net gain seems to be fairly small.

I understand succinctness being a good thing usually but I don't really see any real net gain here.

As for intermediate nil values - to be honest I think it is possible to design methods in such a way as to act as a filter, and discard nil values when they are not necessary to begin with (e. g. making more use of .compact). But as written above, you ultimately have to only convince matz. You can suggest to add it to the next developer discussion to get the opinion(s) of the core devs if you would like to, at:

https://bugs.ruby-lang.org/issues/15129

#3 - 09/22/2018 12:06 AM - jeremyevans0 (Jeremy Evans)

- Backport deleted (2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN)

- Tracker changed from Bug to Feature

koenhandekyn (koen handekyn) wrote:

> proposal allow
>
> ```
> collection.map(&:instrument, &:issuer, &:name)
> ```
>
> implementation is trivial

Is it possible for you to share your trivial implementation of passing more than one block to a method? :)

It is true that the implementation of something like:

```
collection.chained_map(:instrument.to_proc, :issuer.to_proc, :name.to_proc)
```

is fairly trivial, but it could easily be added via an external gem. I don't think it makes sense to add a Enumerable#chained_map core method or to modify the Enumerable#map core method to accept arguments. And I think adding the ability for a method to accept multiple blocks would be another feature request entirely.

In my opinion, it would probably be clearer to just use map with a single block:

```
collection.map { |e| e&.instrument&.issuer&.name }
```