Ruby master - Feature #14390

UnboundMethod#to_proc

01/24/2018 09:18 AM - zverok (Victor Shepelev)

Status:	Feedback	
Priority:	Normal	
Assignee:		
Target version:		

Description

I believe that it could be somewhat useful to have UnboundMethod converted to proc (accepting the object to bind to, as a first argument).

Practical(ish) example, paired with Proc#rcurry proposal:

```
URLS.
  map(&Faraday.method(:get).rcurry[some_get_param: 'value']).
  map(&JSON.method(:parse).rcurry[symbolize_names: true]).
  map(&Hash.instance_method(:dig).rcurry[:foo :bar, :baz])
```

It is somewhat more verbose than a lot of alternative proposals for "shorthand of &method call with arguments", yet requires no changes in parser or language design. With some future shortcuts/operators for #method and #instance_method it can even become pretty short and look like an "idiom".

PS: For the reference, shorthand that was proposed and rejected several times (see #6483, #4146):

```
...map(&:dig(:foo :bar, :baz))
```

As it is indeed looks much shorter than my proposal, it raises a lot of question about what is that :dig(:foo :bar, :baz) and how it should be parsed and whether it can appear outside of &-calls.

History

#1 - 01/24/2018 09:22 AM - zverok (Victor Shepelev)

Ugh, seems it duplicates answer-less #10879.

Though, I'd be happy to raise the priority of discussion higher, and to provide some real-life examples :)

#2 - 01/24/2018 09:31 AM - zverok (Victor Shepelev)

- Description updated

#3 - 01/24/2018 09:31 AM - zverok (Victor Shepelev)

- Description updated

#4 - 01/24/2018 11:55 AM - shevegen (Robert A. Heiler)

> Ugh, seems it duplicates answer-less #10879.

It can probably be referenced to the other report, and your issue here closed (or the other one; I think it does not matter which one).

Some issues are unanswered but this does not necessarily mean that they are not discussed - just sometimes that they may have lower priority or go "under the radar" as a result.

My suggestion would be - if the issue is important to you, and I somewhat agree with you here (primarily because I think it would be nice to be able to have UnboundMethod have to_proc too, simply due to versatility alone, also because it's kind of nice to be able to detach methods, fiddle around with them, then attach them back to something... a bit like LEGO blocks for

kids :D), then you can try to have it registered for the next ruby developer meeting (the one for january already was only some hours ago, so you may have to wait for february; though matz can of course decide pro/con at any moment in time, but he often prefers some discussion for various reasons - see the backtick situation for 3.0).

Anyway, before I write too much, I think it may be better and easier to provide links:

Wiki to keep track of the developer meetings:

https://bugs.ruby-lang.org/projects/ruby/wiki/Main

Latest developer meeting in January 2018:

https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20180124Japan

Some discussion results of that meeting (not sure if everything has been logged but matz has said several things about some of the issues discussed there):

https://docs.google.com/document/d/1q7GaG5KQy5qEKmOAeVucI4CbvYx-nQnSQSfSzyN-9EM/edit

Biggest one is probably when MJIT will be merged. :)

Anyway, if you think that your suggestion is important, I think a ruby core developer may add your issue to the list for the upcoming meeting. That way we may get matz' opinion on it eventually - and "two" issues could be approved or rejected at the same time! :D

(Actually, I think your desscription here is better than the one given in https://bugs.ruby-lang.org/issues/10879)

#5 - 01/24/2018 12:18 PM - zverok (Victor Shepelev)

shevegen (Robert A. Heiler) I am well aware of the process, thanks. In my head, it also includes "ideas are discussed in this bug tracker, then they are added (or not) to next dev.meeting agenda." Do you have a suggestion of the shortcut way of adding it to agenda?

#6 - 05/17/2018 06:50 AM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Feedback

```
map(&Hash.instance_method(:dig).rcurry[:foo :bar, :baz])
```

What's wrong with map{|i| i.dig(:foo :bar, :baz) } which is order of magnitude concise?

#7 - 05/17/2018 11:46 AM - zverok (Victor Shepelev)

shyouhei (Shyouhei Urabe) the point is in gradual enchancing Ruby's approach to functional/declarative chaining.

Let's look at this example:

```
(1..3).each(&method(:print))
(1..3).each { |i| print i }
```

The second, in fact, is shorter, but (at least for me) the first somehow feels more "right" even in this case: for example, because it is more DRY (no "define i → immediately use i"); but also, it forces to structure code flow clearly (e.g. instead of hairballs of large blocks chained, I tend to think about code structuring in "processed by this method, then processed by that method").

Now, when the basic usability of &method(...) is aknowledged by some, when it would be shortened (I hope), it will be an awesome tool of writing short and clean code.

Also, I believe that some room of optimization is present here, e.g. if some proc "knows" (and, at C level, it knows) it is just a Method, then it can be passed and called with less overflow.

Closer to the topic of current issue:

The baseline is this:

```
map{|i| i.dig(:foo :bar, :baz) }
```

And looking at things like map(&:size), lot of people invent things like:

```
.map(&:dig(:foo, :bar, :baz))
# or
.map(&:dig.(:foo, :bar, :baz))
```

...which is interesting but somehow "spheric in a vacuum": what is this syntax? whose responsibility is to maintain it? How should it be read and parsed?

So, what I am trying to think about, is not the necessary shorter code, but conceptually more clean. If the concepts are there, it can be shortened to a special syntax later (like &method(:foo) → &.:foo is planned currently).

So, my idea was (I am not sure about it, but I believe that at least introduction of UnboundMethod#to_proc would not harm anybody or anything) that

```
map(&Hash.instance_method(:dig).rcurry[:foo :bar, :baz])
```

...is not a better way of doing things, just a way that can be made available, to see what optimization and code style techniques can emerge.

It is somehow "logical" to say that .map(&:dig.(:foo, :bar, :baz)) is "rendered" as "call instance method with thouse arguments bound to it".

#8 - 05/18/2018 03:24 AM - shyouhei (Shyouhei Urabe)

Thank you for the explanation. So as far as I understand what you want to do is not just UnboundMethod#to_proc, but a lot more. Seems you are drawing a quite big picture covering structure of code flows. So far I see this UnboundMethod#to_proc feature itself is not that useful, but understand the motivation behind this proposal.