## Optional block argument for `itself`

**09/14/2016 07:39 AM - zverok (Victor Shepelev)**

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
</tbody>
</table>

### Description

That's an another attempt to choose good metaphor for object.(yield self) (previously: [#6721](#6721) and ton of duplicates).

In discussion here: [https://bugs.ruby-lang.org/issues/11717#note-3](https://bugs.ruby-lang.org/issues/11717#note-3) it was said:

> Clearly this is something the Ruby community wants. Just as clearly, it's something nobody can name.

But suddenly... I've thought about this!

```ruby
some.long.method.chain.constructing.string
 .itself ( |s| "(" + s + ")")
 .itself(&method(:puts))

# or
require 'open-uri'

construct_url(*params)
 .itself(&method(:open))
 .read
 .itself(&JSON.method(:parse))
 .to_yaml
 .itself(&File.method(:write))

# NB: I understand that _last_ `itself` in both examples
#     could as well be `tap`. But not all the previous.
```

### Pros:

- method is already here, it has pretty name and underused (almost nothing except group_by(&:itself) comes to mind);
- it is not 100% good English, but readable: itself(&JSON.method(:parse)) = "parse itself with json";
- implementation is trivial, no backwards-compatibility issues (like new methods shadowing something important in third-party library) are expected.

### Cons:

- ???

### Related issues:

<table>
<thead>
<tr>
<th>Is duplicate of</th>
<th>Ruby master - Feature #6721: Object#yield_self</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is duplicate of</td>
<td>Ruby master - Feature #6684: Object#do</td>
<td>Rejected</td>
</tr>
<tr>
<td>Is duplicate of</td>
<td>Ruby master - Feature #7388: Object#embed</td>
<td>Rejected</td>
</tr>
<tr>
<td>Is duplicate of</td>
<td>Ruby master - Feature #10095: Object#as</td>
<td>Closed</td>
</tr>
<tr>
<td>Is duplicate of</td>
<td>Ruby master - Feature #11717: Object#trap -- pass object to b...</td>
<td>Closed</td>
</tr>
<tr>
<td>Has duplicate Ruby master - Feature #13172: Method that yields object to bloc...</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>Has duplicate Ruby master - Feature #13569: Change implementation of Feature ...</td>
<td>Feedback</td>
<td></td>
</tr>
</tbody>
</table>

### History

**#1 - 09/14/2016 10:37 AM - phluid61 (Matthew Kerwin)**

> Sounds familiar, if only someone had submitted a patch or written a gem ;)

But yes, once again, I'm +1 for this proposal. Lots of people have objections, though, as you can see particularly in [#10095](#10095)

---

*03/21/2022*
But, to be completely honest, I'm fascinated by how this simple, easy to implement and useful functionality have drowned for ages in "better-name-ever" discussions.

This is not surprising at all because naming things is one of the hardest things to get "right".

You just have to look at various rubygems that have a very strange name.

Animal names for instance - a webserver called puma? unicorn as a HTTP server for Rack applications?

I can give you another example too, look at various rubygems that have a very strange name. Animal names for instance - a webserver called puma? unicorn as a HTTP server for Rack applications?

People need or should be vaguely comfortable with syntax. Take the lambda stabby operator, I can not overcome my mental obstacle to use it - it just does not "feel right" in my own code. Lots of other people make use of it, that is fine by me.

Back to the topic of .itself, while I have no strong objection, I struggle with other method names such as .tap - whenever I read .tap, I actually think of a certain card game where you have to tap cards. :)

.inject is another odd one, I think of a big nasty syringe (actually my brain associates it more as name like "sum of", because that is what I use .inject most regularly like "array.inject(0){|sum,x| sum + x}"

In conclusion, I think you have to expect a certain resistance in particular for syntax. Best thing is to try to convince matz. And if it won't work within ruby 2.x and neither towards the path towards ruby 3.x and beyond, perhaps it works for ruby 4.x :)

(Actually, perhaps the DSL aspect of ruby could be extended one day to allow sub-languages of ruby a bit like lisp dialects... but let's not get too crazy here.)
Concept of `yield_self` (or whatever we name it) method seems like a huge game-changer to me in a challenge of cleaner and idiomatic code. Yes, it's like Elixir's `|>` operator, but it could be introduced in no effort at all.

Does nobody can see it?
I don't think so, considering how often the question is raised. Four 4 years now (and it is only proposals that are found in tracker)?

OK, let's summarize EVERYTHING that was proposed (and doubted) so far:

- #6721:
  - `yield_self`,
  - `yield_to`,
  - `submit_to`,
  - `surrender`,
  - `capitulate`,
  - `apply`,
  - `ergo` (implemented in Ruby Facets exactly like `#itself` proposed here);
  - `^` (like four = 2 ^ { |x| x*x });
  - `#itself` (again, like `#itself` -- yield-or-return-self)
  - `#cast` (more complicated behavior, like 3.cast(2, :+) => 5)
  - `#itos`
  - `#tap!
- #7388 (Matz had explicitly stated he is against all of the options)
  - `#embed`
  - `#infix`
  - `#ergo again`
- #6684 (proposal & discussion is partially in Japanese)
  - `#do`
- #11717
  - `#trap` (was my idea, I've liked it to be close to `#tap` yet was pointed to confusion with `Signal#trap`)
- #10095
  - `#as`
  - `#chain`
  - `#itself` (and big discussion why it is not appropriate -- don't really persuasive for me, but you can know batter)
  - `#yield`
  - `block form of #send (like (2 + 3).send{ |x| x + 2 })`
  - `new method/operator` ->
  - `Using just | operator`
  - `#continue`

My favourite is, obviously, itself; "reading" reasoning could be found above:

it is not 100% good English, but readable: itself(&JSON.method(:parse)) = "parse itself with json";

But, to be honest, it is not the point. Point is we need it, whatever it is called. And except for "new operator" approach, and versions that will confuse parser (self, yield and do), it is really easy to implement after the name was selected.

Please-please-please, could somebody add this topic to next developer's meeting agenda?..

#6 - 09/19/2016 09:23 AM - duerst (Martin Dürst)
Victor Shepelev wrote:

Please-please-please, could somebody add this topic to next developer's meeting agenda?..

#7 - 09/19/2016 09:49 AM - zverok (Victor Shepelev)
Thanks a lot! (But, sorry for bothering, could it be discussed as "Optional block argument for itself OR another name for yield self method"? Like, in hope for final solution, not "optional block argument for itself is rejected, and that's it". List of proposed names I've gathered in current ticket.)

#8 - 09/20/2016 12:54 AM - nobu (Nobuyoshi Nakada)
- Is duplicate of Feature #6721: Object#yield_self added

#9 - 09/20/2016 12:55 AM - nobu (Nobuyoshi Nakada)
- Is duplicate of Feature #6684: Object#do added

#10 - 09/20/2016 12:55 AM - nobu (Nobuyoshi Nakada)
- Is duplicate of Feature #7388: Object#embed added

#11 - 09/20/2016 12:55 AM - nobu (Nobuyoshi Nakada)  
- Is duplicate of Feature #10095: Object#as added

#12 - 09/20/2016 12:55 AM - nobu (Nobuyoshi Nakada)  
- Is duplicate of Feature #11717: Object#trap -- pass object to block and return result added

#13 - 09/20/2016 12:56 AM - nobu (Nobuyoshi Nakada)  
What about let ?:

```
some.long.method.chain.constructing.string
  .let { |s| "(* + s + *)" }
```

#14 - 10/11/2016 10:33 AM - shyouhei (Shyouhei Urabe)  
FYI we looked at this issue at developer meeting today. Because matz was not there this extension still has chance, but the attendees thought itself's block (if any) shall work like how tap works. So it might not be suitable for the requested functionality.

#15 - 10/11/2016 11:07 AM - zverok (Victor Shepelev)  
Thanks for the update! Haven't you discussed the probable another name for it?.. We just want it to be in language already.

#16 - 11/02/2016 07:07 AM - knu (Akinori MUSHA)  
What about giving it an obvious name like yield_self and making a syntax sugar object.[|x| ... ] for calling it like we do in Proc#call?

#17 - 11/25/2016 08:10 AM - matz (Yukihiro Matsumoto)  
yield_self is OK, but I don't think we are going to add object.[|x| ... ].

Matz.

#18 - 11/25/2016 08:43 AM - zverok (Victor Shepelev)  
Well, I should say yield_self is not very readable in context (it raises question which is "self" in that context?):

```
construct_url(*params)
  .yield_self(&:method(:open))
  .read
  .yield_self(&:JSON.method(:parse))
  .to_yaml
  .yield_self { |yml| File.write('cache.yml', yml) }
```

But I'd rather prefer to finally have this method in core language, whatever it will be named :)

#19 - 01/31/2017 04:16 AM - nobu (Nobuyoshi Nakada)  
- Has duplicate Feature #13172: Method that yields object to block and returns result added

#20 - 02/06/2017 08:54 AM - Nondv (Dmitry Non)  
Hello.  
Is there any movement?

#21 - 04/17/2017 06:47 AM - ko1 (Koichi Sasada)  
how about alter?  
at dev meeting, someone said reform.

#22 - 04/17/2017 07:36 AM - zverok (Victor Shepelev)  
I believe the situation should be addressed just by somebody's authority.  
What do you think about appointing the name selection for next core dev meeting? You can just take a list that gathered in current ticket and do some voting or something.  
I don't think it is an acceptable situation when really useful method is not introduced for 4 years since first proposal (if it haven't proposed before, which I am not sure) just because nobody wants to choose the name.  
TBH, for me open("http://some.url").read.itself(&:JSON.method(:parse)) still looks like a cleanest solution possible, yet I'd just prefer to see this method in core, with whatever name.
If Matz is OK with yield_self, let it be yield_self.

#23 - 05/01/2017 07:50 AM - nobu (Nobuyoshi Nakada)
- Status changed from Open to Closed

Applied in changeset trunk|r58528.

object.c: Kernel#yield_self
  - object.c (rb_obj_yield_self): new method which yields the receiver and returns the result. [ruby-core:46320] [Feature #6721]

#24 - 05/13/2017 02:24 PM - nobu (Nobuyoshi Nakada)
- Has duplicate Feature #13559: Change implementation of Feature #6721 added