Ruby trunk - Feature #14594

Rethink yield_self's name

03/09/2018 09:46 AM - zverok (Victor Shepelev)

Status:	Closed
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	

Description

I feel really uncomfortable raising the question again, but...

In several months since 2.5 release I've written a lot of code with yield_self (using backports gem with earlier versions of Ruby when necessary), and explained it several times to students, and colleagues (and in this blog post which have gained pretty decent attention).

I should say that I am still assured the name chosen is really not optimal. Reasons:

- it is just too long for such a basic operation;
- it does not say "what it does", but rather "how it is implemented"; it is like having each_returning_block_result instead of map;
- self is really misguiding and obscure in situations like this:

```
class MyClass
  def some_method
    @path.yield_self(&File.method(:read)).yield_self(&Parser.method(:new)) ...
  end
end
```

Intuitively, word "self" inside instance method is read like it somehow related to current context's self (e.g. instance of MyClass), which it is absolutely not. In other words, "self" in caller's context has nothing to do with "self" implied by method's name.

After reconsidering a lot of options, **my current proposal is: #then**.

Reasons:

- despite being a keyword, something.then(something) is not a conflicting Ruby syntax, and allowed by current Ruby;
- it is short!
- it shows intention pretty well, and reads natural, in both cases: when receives block and when returns Enumerator:

```
File.read(filename).then(&JSON.method(:parse))
rand(10).then.detect(&:odd?)
```

In many languages, .then or .and_then is useful construct, meaning the same (calculate next value from the result of the previous operation), just in a narrower context of futures/promises. I believe that even when/if Ruby will have those as a language feature, that syntax will play well:

```
value.then(&:computation) # => value
promise.then(&:computation) # => promise
```

PS: For historical reasons, here is huge list of previous proposals I've gathered for this method name.

Associated revisions

Revision d53ee008 - 05/30/2018 08:24 AM - matz (Yukihiro Matsumoto)

object.c: Add a new alias then to Kernel#yield_self; [Feature #14594]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63525 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 63525 - 05/30/2018 08:24 AM - matz (Yukihiro Matsumoto)

object.c: Add a new alias then to Kernel#yield_self; [Feature #14594]

Revision 63525 - 05/30/2018 08:24 AM - matz (Yukihiro Matsumoto)

object.c: Add a new alias then to Kernel#yield_self; [Feature #14594]

Revision b3fb892d - 07/01/2018 08:12 AM - ktsj (Kazuki Tsujimoto)

NEWS: add NEWS entry about Kernel#then [Feature #14594]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63816 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 63816 - 07/01/2018 08:12 AM - ktsj (Kazuki Tsujimoto)

NEWS: add NEWS entry about Kernel#then [Feature #14594]

Revision 63816 - 07/01/2018 08:12 AM - ktsj (Kazuki Tsujimoto)

NEWS: add NEWS entry about Kernel#then [Feature #14594]

History

#1 - 03/09/2018 01:50 PM - shevegen (Robert A. Heiler)

I agree to your statement that yield_self is not a good, descriptive name.

Though I have to admit that I never used yield_self so far. I can not even say what it does, either. :)

I like yield and self. I don't like the name yield_self so I could not use it so far. I use only a subset of ruby that makes sense to me and I find appealing. Thankfully one can use ruby just fine without having to use everything.

> After reconsidering a lot of options, my current proposal is: #then.

I dislike that as well. I also do not think that "then" makes a lot of sense.

For example:

```
File.read(filename).then(&JSON.method(:parse))
```

This is regular method chaining. But it sounds like you are using a conditional there.

"then" is already a keyword in ruby isn't it?

https://docs.ruby-lang.org/en/2.5.0/keywords_rdoc.html

I think it is confusing to have keywords and method names, even more so when they do different things.

> In many languages, .then or .and_then is useful construct, meaning the same (calculate next value from the result of the previous operation), just in a narrower context of futures/promises.

Realistically ruby should strive for intrinsic consistency first, not for what features or anti-features other languages may or may not have.

> I believe that even when/if Ruby will have those as a language feature, that syntax will play well:

```
value.then(&:computation) # => value
promise.then(&:computation) # => promise
```

I don't think it reads nicely really.

yield_self is not a good name but your proposal is also not good, in my opinion. But it's just a personal opinion, feel free to ignore it. At the end of the day you only have to convince matz. :)

The name yield_self is however had indeed not a good name.

Finding good names is quite difficult. Single words are also almost always better than combined names, even though one is a bit limited with single words alone. There are exceptions though. For example **.each_with_index** or **.each_index** are good names, IMO.

#2 - 03/23/2018 08:08 PM - cmoel (Christopher Moeller)

I also agree yield_self isn't the best name but I'm also skeptical of then. Why not use map, which we already have for collections? Generally speaking, the map function isn't just about collections (though that's usually how it's used in Ruby). map is more about putting an object in a context (a block in Ruby's case), modifying the object, and returning the modified object.

Another benefit to calling the method map instead of yield_self or then is most Rubyists already understand map and how it's applied to collections, so there probably wouldn't be a large learning curve for it. It can be summarized like this: "Object#map is similar to applying map to an array containing 1 element."

Though I'm not familiar with the details, Enumerable#map and friends' implementation of map could then be an override of Object#map, yielding the block to each element in the collection, which would maintain how map currently works on collections.

Can anyone think of any issues with using map as a different/better name for yield_self?

#3 - 03/23/2018 08:16 PM - zverok (Victor Shepelev)

> Why not use map

```
paragraphs
  .map { something }
  .reject { something }
  .yield_self { |ps| external_post_processor(ps, **options) }
```

#4 - 03/25/2018 07:23 AM - sawa (Tsuyoshi Sawada)

map is one of the worst names for this feature in my opinion. Generalizaing from cmoel's proposal to use map for this feature, we expect:

```
[1].map{|e| e * 2} #=> [1, 1]
```

but that is not what we currently have, nor does cmoel seem to want that. Instead, we have, and cmoel seems to want to handle as an exceptional case, the following:

```
[1].map{|e| e * 2} #=> [2]
```

That is confusing. Why do you want to make something that has worked without problems to this day to become an exception?

#5 - 03/25/2018 07:32 AM - sawa (Tsuyoshi Sawada)

I disagree with the names proposed, but I agree that the current name is too long.

Here are yet other name candidates from me:

- deform
- expand
- reform

Edit. I just realized that reform has already been mentioned. Sorry.

#6 - 03/28/2018 02:15 AM - bughit (bug hit)

then does seem better but you have to keep in mind you'd be barring this name from all present and future domain specific uses in libraries and apps

What about tap! - since tap is already in use, this won't be stealing another word, and it seems to make sense, it's a "dangerous" tap that alters the flow.

#7 - 04/02/2018 02:31 AM - jeromedalbert (Jerome Dalbert)

I agree that yield_self is not a great name. But then does not "say what it does" enough in my opinion. I prefer itself: this name has problems too, but it is more descriptive at least.

#8 - 04/02/2018 06:50 AM - zverok (Victor Shepelev)

> I prefer itself

itself is already taken in block-less form.

It was my initial idea too (see links in the post for names discussion), but I am happy it was not accepted: iterator-returning block-less yield_self turns out to be useful too.

#9 - 04/02/2018 01:13 PM - cmoel (Christopher Moeller)

bughit (bug hit) wrote:

> What about tap! - since tap is already in use, this won't be stealing another word, and it seems to make sense, it's a "dangerous" tap that alters the flow.

I think tap! is the best option currently being discussed:

1. There's no need to create another word for this kind of generic functionality,
2. Rubyists already know and use tap, and
3. Rubyists also know the ! suffix modifies the callee.

#10 - 04/05/2018 05:12 PM - americodls (Americo Duarte)

My two cents:

This method acts like a **pipe operator**, it passes itself **through** the block and returns whatever the block returns.
The block execution maybe will change the value itself, maybe not. But yield_self proposes **chaining** changes without mutation (tap is recommended to make mutations since tap returns the receiver itself instead of the block result).

So I think good candidates to rename or alias yield_self:

- pipe
- pipe_to
- chain
- chain_with.

#11 - 04/09/2018 10:45 AM - BatuhanW (Batuhan Wilhelm)

I think expand is cool, it is expanding method to another one, so it can be chained.

My alternative suggestion would be "pass" since it is passing itself. What do you think about it?

```
construct_url
  .pass(&Faraday.method(:get)).body
  .pass(&JSON.method(:parse))
  .dig('object', 'id')
  .pass { |id| id || '<undefined>' }
  .pass { |id| "server:#{id}" }
```

#12 - 04/18/2018 04:03 AM - sowieso (So Wieso)

I just like to stress that having an easy and short name is more important than having a name that correctly describes the behaviour. *tap* is also a horrible name that every ruby newbie has to look up first, still it is loved in the community. And that's especially due to the fact that it is short and feels lightweight. As for yield_self, no discussion is needed whether this method is needed, but the naming is indeed the difficult part. So I support this discussion (as annoying as it is, how many years is it now?). Let's find a better name (probably an alias)!

I would go with (in this order):

- **tip** (the closest relative to tap, absolutely short, *tip over* suggests some movement/transfer, probably never used anywhere, tip-tap sounds like a cat walking, downside: not very descriptive at all)
- and (like then, even shorter, focus on continuing, probably completely unused, downside: people might think of boolean comparisons, not very descriptive)
- then (short, downside: people might think of promises, not very descriptive)
- pass (short and somewhat descriptive, might be in use already somewhere)
- pipe (short and more descriptive, but used in other contexts (threads, shell))

Some other ideas:

- hand (as in hand over, short, probably never used elsewhere, descriptive? (with some imagination))
- chain (probably in use too often)
- use (use_here)
- pass_me (somewhat longer, pretty clear, reads much nicer than yield_self, but I don't like the *me* either)
- yield (already in use, so a no-go, without the self it's short, similar to pass, but too confusing I guess)
- relay (probably in use too often)

I don't like tap!, I think it's misleading, as I would guess the method receiver would get mutated. map is way to confusing. Including *self* inside a

method name is always somewhat confusing (self in which context?), so I would prefer not to do it.

So many ideas, there are definitely a lot I would be content with, we just need to decide.

#13 - 04/18/2018 10:52 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

I actually liked the tip suggestion.

#14 - 04/19/2018 06:43 AM - matz (Yukihiro Matsumoto)

- Assignee set to matz (Yukihiro Matsumoto)

After a long consideration, I decided to pick then as an alias to yield_self.
As zverok (Victor Shepelev) stated it describes intention, not behavior.

Because then is a reserved word, it has some restriction, but I think it is acceptable.

Matz.

#15 - 05/30/2018 08:25 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Closed

Applied in changeset trunk|r63525.

---

object.c: Add a new alias then to Kernel#yield_self; [Feature #14594]

#16 - 05/31/2018 01:01 PM - jrochkind (jonathan rochkind)

I'm concerned that then is used in Promises, and is already in use in many promise-implementing libraries, including concurrent-ruby.

The Promise libraries could change the name of their method to not conflict, but then it's not only backwards compatibility but a mis-match with Promises in other platforms, and the original JS spec that they are based on.

The Promise libraries can keep doing this, effectively overriding the Kernel method. Somewhat confusingly then will mean something different when used on a Promise object, and Kernel#then won't be available on them. I guess since it's just an alias, this isn't a disaster, yield_self is still available, but if you're using Promises you've got to think about what kind of object you have and which alias you want, and that then means something different on Promises.

#17 - 05/31/2018 04:50 PM - matz (Yukihiro Matsumoto)

jrochkind (jonathan rochkind) It is introduced that a normal object can behave like promises.
So the name conflict is intentional.
If you really wanted a non-unwrapping method for promises, use yield_self.

Matz.

#18 - 05/31/2018 11:08 PM - avit (Andrew Vit)

matz (Yukihiro Matsumoto) wrote:

> If you really wanted a non-unwrapping method for promises, use yield_self.

If I understand what you mean by "unwrapping" here, the new method still doesn't call yielded procs to make them composable: it's only an alias for yield_self, right?

Is this still a possible consideration?

- https://bugs.ruby-lang.org/issues/6284
- https://bugs.ruby-lang.org/issues/13600

Thanks for this by the way, I very much prefer the new name!

#19 - 05/31/2018 11:17 PM - matz (Yukihiro Matsumoto)

Right. I am not against the idea of function composition. But it should be discussed separately from yield_self and then. Remember I rejected #13600 but not #6284.

Matz.

#20 - 06/07/2018 06:49 PM - avastor.developer (Avastor Avastor)

```
a = 5
y = if a.then then 1 else 2 end
```

Hmm. What are you trying do with Ruby?

#21 - 09/02/2018 02:35 PM - RichOrElse (Ritchie Buitre)

I suggest aliasing #yield_self with **#to_be**. The to_ prefix suggest returning a value while the word "be" connotes both identity (#itself) and it's new condition (&block).

Not only is it shorter, when reading it repeatedly, I also don't find it annoying. It reads just like the lyrics to the song "Welcome to My Life" by Simple Plan.

```
construct_url.
  to_be(&Faraday.method(:get)).body.
  to_be(&JSON.method(:parse)).dig('object', 'id').
  to_be { |id| id || '<undefined>' }.
  to_be { |id| "server:#{id}" }
```

#22 - 09/04/2018 02:48 AM - sowieso (So Wieso)

This looks **to_be** perfect, I really like this idea!

#23 - 09/04/2018 12:21 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

I liked to_be as well.

#24 - 09/04/2018 12:22 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

And I like pipe too, maybe even more.

#25 - 10/05/2018 04:55 PM - theorygeek (Ryan Foster)

I think then is going to be a bad name, because it has a special meaning in A+ Promises: https://promisesaplus.com/

This makes every object in Ruby a "thenable" object, and I think it will be bad for promise libraries.

#26 - 12/14/2018 03:30 AM - sowieso (So Wieso)

So we will have the questionable (as confusing for promise library users) then in 2.6? There were a lot of concerns mentioned above. to_be (or other suggestions) wouldn't have these problems. I would advice to re-think this one final time, as we can't change it anymore later…

#27 - 12/14/2018 03:50 AM - mame (Yusuke Endoh)

FYI: mruby 2.0 has been already released with Kernel#then. I think we've gone beyond the point of no return.

#28 - 01/09/2019 10:15 AM - WaKeMaTTa (Mohamed Ziata)

I think the word should be hit.

Example:

```
construct_url.
  hit(&Faraday.method(:get)).body.
  hit(&JSON.method(:parse)).dig('object', 'id').
  hit { |id| id || '<undefined>' }.
  hit { |id| "server:#{id}" }
```