

## Ruby master - Feature #6373

### public #self

04/28/2012 06:46 AM - trans (Thomas Sawyer)

<b>Status:</b>	Closed	
<b>Priority:</b>	Normal	
<b>Assignee:</b>	matz (Yukihiro Matsumoto)	
<b>Target version:</b>		
<b>Description</b>		
This was recently suggested to me as an extension:		
<pre>class Object   # An identity method that provides access to an object's 'self'.   #   # Example:   #   [1, 2, 3, 4, 5, 1, 2, 2, 3].group_by(&amp;:identity)   #   #=&gt; {1=&gt;[1, 1], 2=&gt;[2, 2, 2], 3=&gt;[3, 3], 4=&gt;[4], 5=&gt;[5]}   #   def identity     self   end end</pre>		
First, is such a method commonly useful enough to warrant existence?		
Second, it makes me wonder if #self should be a public method in general.		
<b>Related issues:</b>		
Related to Ruby master - Feature #6817: Partial application		<b>Open</b>

### Associated revisions

#### Revision 0a0160d6 - 08/02/2014 01:26 AM - nobu (Nobuyoshi Nakada)

object.c: Object#itself

- object.c (rb\_obj\_itself): new method Object#itself. based on the patch by Rafael França in [ruby-core:64156]. [EXPERIMENTAL] this method may be renamed due to compatibilities. [ruby-core:44704] [Feature #6373]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47028 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

#### Revision 47028 - 08/02/2014 01:26 AM - nobu (Nobuyoshi Nakada)

object.c: Object#itself

- object.c (rb\_obj\_itself): new method Object#itself. based on the patch by Rafael França in [ruby-core:64156]. [EXPERIMENTAL] this method may be renamed due to compatibilities. [ruby-core:44704] [Feature #6373]

#### Revision 47028 - 08/02/2014 01:26 AM - nobu (Nobuyoshi Nakada)

object.c: Object#itself

- object.c (rb\_obj\_itself): new method Object#itself. based on the patch by Rafael França in [ruby-core:64156]. [EXPERIMENTAL] this method may be renamed due to compatibilities. [ruby-core:44704] [Feature #6373]

#### Revision 47028 - 08/02/2014 01:26 AM - nobu (Nobuyoshi Nakada)

object.c: Object#itself

- object.c (rb\_obj\_itself): new method Object#itself. based on the patch by Rafael França in [ruby-core:64156]. [EXPERIMENTAL] this method may be renamed due to compatibilities. [ruby-core:44704] [Feature #6373]

#### Revision 47028 - 08/02/2014 01:26 AM - nobu (Nobuyoshi Nakada)

object.c: Object#itself

- object.c (rb\_obj\_itself): new method Object#itself. based on the patch by Rafael França in [ruby-core:64156]. [EXPERIMENTAL] this method may be renamed due to compatibilities. [ruby-core:44704] [Feature #6373]

## Revision 47028 - 08/02/2014 01:26 AM - nobu (Nobuyoshi Nakada)

object.c: Object#itself

- object.c (rb\_obj\_itself): new method Object#itself. based on the patch by Rafael França in [ruby-core:64156]. [EXPERIMENTAL] this method may be renamed due to compatibilities. [ruby-core:44704] [Feature #6373]

## Revision 47028 - 08/02/2014 01:26 AM - nobu (Nobuyoshi Nakada)

object.c: Object#itself

- object.c (rb\_obj\_itself): new method Object#itself. based on the patch by Rafael França in [ruby-core:64156]. [EXPERIMENTAL] this method may be renamed due to compatibilities. [ruby-core:44704] [Feature #6373]

## History

---

### #1 - 04/28/2012 07:11 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

### #2 - 04/28/2012 08:28 AM - duerst (Martin Dürst)

#identity (or whatever it's called) is quite important in functional languages. It's handy to pass to another function that e.g. uses it as an argument to map.... So I think it's a good idea to add it.

The name should be identity or id (Haskell) or some such.

Making self public sounds impressive, but first, it doesn't expose anything new (it's the object itself, which is already accessible) an second, I feel it's the wrong name, because #identity essentially only makes sense in locations in a program where the actual object isn't directly around.

### #3 - 04/28/2012 10:05 AM - naruse (Yui NARUSE)

Use `id`.

### #4 - 04/28/2012 04:02 PM - headius (Charles Nutter)

At first I found this laughable, but then I realized there's no clear method you can call against an object that simply returns the object. It is a small thing, but turns out to be very useful.

For example, if you want a chain of method calls that successively transformation some data. Those transformations, in the form of folds or filters or what have you, may accept an object and return some new object by calling a method. In that case, the simplest transformation is to simply return the object unmodified. There is no such method on Object or BasicObject right now, so rather than having the uniformity of a method call you would need a special filter form that calls nothing and returns its one argument. Identity is an easy, simple functional form that should be available.

I would suggest it should be called `self` or similar, since it is a key core method nobody should ever override. But I do think there is utility in having it always available.

### #5 - 04/28/2012 04:06 PM - headius (Charles Nutter)

Actually, I just realized there is already a method you can call to just return the method itself, albeit in a really gross way: `#tap`.

```
obj.tap{} #returns obj; block is required and invoked
```

### #6 - 04/28/2012 06:31 PM - prijutme4ty (Ilya Vorontsov)

`enum.map(&:identity)` can be replaced with `enum.to_a`

But I think `Object#identity` is useful. `Object#tap` requires a block. It's not an option to use it in many cases. I can't imagine how to replace `enum.group_by(&:identity)` with `tap` instead of `identity`.

I think that `tap` method can be improved to be used without block yielding self. But for readability it'd be however aliased to `identity`.

`__self__` looks like a method every programmer'd avoid.

### #7 - 04/29/2012 12:10 AM - matz (Yukihiro Matsumoto)

`__id__` returns `object_id` number, `identity` here is supposed to return itself.

I agree with introducing method to return self, but not fully satisfied with the name 'identity'.

Any opinion?

Matz.

### #8 - 04/29/2012 12:53 AM - trans (Thomas Sawyer)

Public `#self` seems like the most obvious choice. Is there some reason not to use it?

**#9 - 04/29/2012 12:59 AM - regularfry (Alex Young)**

On 28/04/2012 16:10, matz (Yukihiro Matsumoto) wrote:

Issue [#6373](#) has been updated by matz (Yukihiro Matsumoto).

`__id__` returns `object_id` number, `identity` here is supposed to return itself.  
I agree with introducing method to return self, but not fully satisfied with the name 'identity'.  
Any opinion?

"itself"?

--

Alex

**#10 - 04/30/2012 11:20 PM - pabloh (Pablo Herrero)**

What about if we borrow `#yourself` message name from Smalltalk?

**#11 - 05/01/2012 01:10 AM - alexeymuranov (Alexey Muranov)**

Thomas, i think an argument against public `#self` is that 'self' is a reserved word, which moreover is used more as an object name than as a method name. So 'self' would need to stop being a keyword and become a public 'predefined' method of BasicObject (it cannot be defined without the 'self' keyword, i guess).

I like `#itself` or `#yourself`, but i do not know which one is a proper way to talk to my objects.

**#12 - 05/01/2012 02:03 AM - trans (Thomas Sawyer)**

Like many of Ruby's keywords, it can still be used to define a public method:

```
class X
  def self; "x"; end
end
```

```
x = X.new
x.self #=> "x"
```

**#13 - 05/08/2012 10:16 PM - shevegen (Robert A. Heiler)**

Perhaps Smalltalk has the best suggestion. :)

**#14 - 05/15/2012 03:41 AM - alexeymuranov (Alexey Muranov)**

Another option: `#the_self`. The same number of symbols as in `#yourself`, but harder to type :(.

**#15 - 05/15/2012 03:59 AM - Eregon (Benoit Daloze)**

On 28 April 2012 17:54, Alex Young [alex@blackkettle.org](mailto:alex@blackkettle.org) wrote:

"itself"?

I agree, `#itself` is the best to me.

**#16 - 06/24/2012 03:34 AM - marcandre (Marc-Andre Lafortune)**

I second the addition of `Object#self`.

For the objection that `self` is a keyword, so is `class`. And there wouldn't ever be a need to call `self.self` :-)

Do we need a slide-show for this?

**#17 - 07/03/2012 04:15 AM - trans (Thomas Sawyer)**

Does it really need a slide? Does someone at developers meeting want to bring it up? It's such a simple thing. It's probably a one line addition to code to make `self` available as public method. Only question is name, which seems to me, why have different public name than private name? Go with `#self`. But if necessary for public/private names to differ, everyone seems okay with `#itself`.

**#18 - 07/11/2012 12:38 AM - alexeymuranov (Alexey Muranov)**

I've heard that the underscore `_` is commonly used for ignored block variables. Maybe this "public self" can be considered as an "ignored method", and called `Object#_?`

**#19 - 07/11/2012 04:37 AM - trans (Thomas Sawyer)**

`__` is used by irb. Also, I don't really see why. Code would look much more "perlish" using `__`.

Please, what's wrong with public `#self`?

**#20 - 07/11/2012 05:55 AM - alexeymuranov (Alexey Muranov)**

trans (Thomas Sawyer) wrote:

Please, what's wrong with public `#self`?

Nothing, just was wondering how to use `__` :). I didn't know IRB uses it.

**#21 - 07/30/2012 03:42 AM - citizen428 (Michael Kohl)**

FWIW, I'm the one who suggested this method as an addition to Facets, mainly for the reason headius mentions above, it's the simplest filter available. I'm still torn on the name, but for some reason `#self` didn't seem right. For my own extension library I finally went with `#it`.

**#22 - 08/07/2012 02:47 AM - citizen428 (Michael Kohl)**

i also found a previous issue where this behavior would come in handy: <http://bugs.ruby-lang.org/issues/2172>

**#23 - 08/07/2012 06:00 AM - alexeymuranov (Alexey Muranov)**

How about merging this with feature request [#6721](#) for `#yield_self`?

Object`#self` can optionally accept a block, yield self to the block if block given, and return the result of the block. What do you think?

**#24 - 08/07/2012 06:20 AM - trans (Thomas Sawyer)**

Strikes me as a very good idea! I forgot about that. In Facets it is called `#ergo`. Essentially,

```
def ergo
  return yield(self) if block_given?
  self
end
```

Call it `#self` instead and we get two features for the price of none!

Not sure if should take additional `*args` or not, but it could.

**#25 - 08/10/2012 02:22 PM - Anonymous**

matz (Yukihiro Matsumoto) wrote:

`__id__` returns object\_id number, identity here is supposed to return itself.  
I agree with introducing method to return self, but not fully satisfied with the name 'identity'.  
Any opinion?

Matz.

I did some thinking, and there is hardly anything better than 'identity'.  
'identity\_function' would be hypercorrect, and 'id' is commonly understood as identifier. In my personal library, I use 'i' (small cap Unicode I) for identity mapping, though...

**#26 - 08/10/2012 04:47 PM - alexeymuranov (Alexey Muranov)**

boris\_stitnicky (Boris Stitnicky) wrote:

matz (Yukihiro Matsumoto) wrote:

`__id__` returns object\_id number, identity here is supposed to return itself.  
I agree with introducing method to return self, but not fully satisfied with the name 'identity'.  
Any opinion?

Matz.

I did some thinking, and there is hardly anything better than 'identity'.  
'identity\_function' would be hypercorrect, and 'id' is commonly understood as identifier. In my personal library, I use 'i' (small cap Unicode I) for identity mapping, though...

ID as an identifier or a piece of identification and *Id* as the *identity function* are two different meanings, as far as i understand. Also it does not look to me like "method" and "function" have exactly the same semantics.

**#27 - 08/10/2012 11:53 PM - Anonymous**

On Fri, Aug 10, 2012 at 4:25 PM, matz (Yukihiro Matsumoto) [m...@ruby-lang.org](mailto:m...@ruby-lang.org) wrote:

`__id__` returns object `id` number, `identity` here is supposed to return itself.  
I agree with introducing method to return self, but not fully satisfied with the name 'identity'.  
Any opinion?

How about allowing `Object#tap` to take no block, simply returning self?  
This syntactic sugar would allow

```
this.tap
```

instead of

```
this.tap {}
```

If a new method must be added, please for the love of sanity don't call it `id` or `identity` or `identifier`. This space already chronically overloaded by frameworks, the need for entities to have publically visible `identity` notwithstanding.

Perhaps `__object__`, which would feel natural to anyone already familiar with `__id__`? Failing that, `Object#object`, although I'd expect this to break a fair amount of existing code. :-)

Ciao,  
Sheldon.

**#28 - 08/11/2012 05:17 AM - trans (Thomas Sawyer)**

Why no answer for: "Why not just public `#self`"? Why add YAMS?

(YAM = Yet Another Method)

**#29 - 10/31/2012 02:23 AM - matz (Yukihiro Matsumoto)**

- Status changed from Assigned to Feedback
- Target version changed from 2.0.0 to 2.6

The point is when we see the code like:

```
[1,2,3,4,5,1,2,2,3].group_by(&:self)
```

sometimes it would be less intuitive that `self` refers elements in the array, not `self` in the scope.  
I think this cause YAM syndrome here. We haven't met name consensus yet (as usual), so I postpone to next minor.

Matz.

**#30 - 06/06/2013 02:45 AM - laise (Alexey Chernenkov)**

Quote: "I think that `tap` method can be improved to be used without block yielding `self`."

+1

It is a VERY usefull feature! Can't understand why `#tap` still need to be used with block only.

**#31 - 06/06/2013 06:54 AM - phluid61 (Matthew Kerwin)**

laise (Alexey Chernenkov) wrote:

Quote: "I think that `tap` method can be improved to be used without block yielding `self`."

+1

It is a VERY usefull feature! Can't understand why `#tap` still need to be used with block only.

Because it's called "tap." Tap doesn't "return self", it taps into an execution flow, extracting an intermediate value for inspection without interrupting the original flow. The analogy is literally tapping a hole in a pipe, to extract liquid samples at various phases in a process. (Also think of tapping a phone line.) Changing its semantics to a straight-up "returns self" method would just make it idiosyncratic, instead of metaphoric.

Matz wrote:

The point is when we see the code like:

```
[1,2,3,4,5,1,2,2,3].group_by(&:self)
```

sometimes it would be less intuitive that self refers elements in the array, not self in the scope.

I have trouble imagining such a scenario. I actually think the some\_array.group\_by(&:self) example is a strong plus for this feature.

I'm also not entirely convinced it's really YAM, since 'self' is already a word in the language, all we're doing is pushing it from keyword to keyword+method, i.e. making it more easily accessible from outside the instance. Even more so since this use-case is really the only place it will ever show up; if there was another way to to\_proc the 'self' keyword to make it easy to pass to a method like group\_by I'd be for that as well/instead.

**#32 - 06/07/2013 03:06 PM - Anonymous**

trans (SYSTEM ERROR) wrote:

Why no answer for: "Why not just public #self"? Why add YAMS?

(YAM = Yet Another Method)

Because explicit identity element is a VIP of the functional space. Summary of the discussion thus far + my opinions follow:

Proposal:	My opinion:	Remark:
#self	+1	proposed and defended by OP, objected by Matz
#identity	+1	strong objection by Anonymous
#_id_	0	
#_self_	-1	
#yourself	-1	
#itself	+3	
#_	-1	
#the_self	-1	
#it	-1	
#_object_	-1	
#id	-1	not proposed, Anonymous strongly objects this option
#tap	-2	me and phluid61 strongly object

I used to favor #identity, but today, I favor #itself. I already advertised this method, and #ergo method ([#6721](http://stackoverflow.com/questions/16932711)) on <http://stackoverflow.com/questions/16932711>, where these features are in demand. To make the naming decision more difficult, let me append Freudian #ego to the list, which would go nicely with #ergo.

**#33 - 06/24/2013 03:51 AM - headius (Charles Nutter)**

phluid61 (Matthew Kerwin) wrote:

laise (Alexey Chernenkov) wrote:

It is a VERY usefull feature! Can't understand why #tap still need to be used with block only.

Because it's called "tap." Tap doesn't "return self", it taps into an execution flow, extracting an intermediate value for inspection without interrupting the original flow. The analogy is literally tapping a hole in a pipe, to extract liquid samples at various phases in a process. (Also think of tapping a phone line.) Changing its semantics to a straight-up "returns self" method would just make it idiosyncratic, instead of metaphoric.

I still like #tap.

1. It would only require removal of the block-yielding requirement.
2. It's functionally equivalent to tapping into execution flow but doing nothing, as in tap {}. I think the objection that it no longer means we're tapping execution flow is a bit pedantic; we *are* tapping execution flow, but the only operation we seek is the reference to the object.

My favorite remains #self, but I can appreciate objections, especially confusion over "self" and "obj.self" and especially "self.self" if #self can be overridden. It does have the advantage that there's probably fewer libraries that define their own "self" method than any other suggestion here.

Other systems I know would use #identity. It also maps to functional programming language for a function that just returns its sole argument (in this case, the 0th argument, the object itself). I would support #identity as well.

Matz wrote:

The point is when we see the code like:

```
[1, 2, 3, 4, 5, 1, 2, 2, 3].group_by(&:self)
```

sometimes it would be less intuitive that self refers elements in the array, not self in the scope.

I have trouble imagining such a scenario. I actually think the some\_array.group\_by(&:self) example is a strong plus for this feature.

I'm also not entirely convinced it's really YAM, since 'self' is already a word in the language, all we're doing is pushing it from keyword to keyword+method, i.e. making it more easily accessible from outside the instance. Even more so since this use-case is really the only place it will ever show up; if there was another way to to\_proc the 'self' keyword to make it easy to pass to a method like group\_by I'd be for that as well/instead.

Another option based on matz's objection: #reference. We want a method that returns the reference to the object we're calling against. #reference seems logical.

```
[1, 2, 3, 4, 5, 1, 2, 2, 3].group_by(&:reference)
```

Variations on this might be #self\_reference, #self\_ref, #selfref. Also #self\_object, #self\_obj, #selfobj.

#self and #identity are probably the most likely to be guessed by a new user.

#### #34 - 06/24/2013 04:07 AM - headius (Charles Nutter)

Another argument why "tap" is fine...

If tap were defined in a functional style, it would be simply

```
def tap(obj, &block)
  block.call(obj)
  obj
end
```

Anywhere you can pass a function you should be able to pass a no-op function, so tap could be defined as

```
def tap(obj, &block)
  block = proc{} unless block
  block.call(obj)
  obj
end
```

So defining tap such that it defaults to a no-op function (i.e. does not yield if block not given) seems perfectly valid to me.

#### #35 - 06/24/2013 09:02 AM - phluid61 (Matthew Kerwin)

headius (Charles Nutter) wrote:

Another option based on matz's objection: #reference. We want a method that returns the reference to the object we're calling against. #reference seems logical.

```
[1, 2, 3, 4, 5, 1, 2, 2, 3].group_by(&:reference)
```

+1. It's sensible (i.e. anyone who knows OOP knows what 'reference' means), there's no overloading of names, and the intention is clear.

So defining tap such that it defaults to a no-op function (i.e. does not yield if block not given) seems perfectly valid to me.

I know I'm throwing a lot of paint at this shed, but while I agree that a default noop #tap is valid, I still strongly believe it makes ary.group\_by(&:tap)

seem like voodoo. I like #reference a lot.

**#36 - 06/24/2013 09:37 AM - charliesome (Charlie Somerville)**

I think out of all the options proposed, 'identity' is the most readable/quickly understandable.

For example, I think the use of 'identity' reads very nicely in `[1,2,3,4].group_by(&:identity)`

**#37 - 06/24/2013 11:07 AM - phluid61 (Matthew Kerwin)**

charliesome (Charlie Somerville) wrote:

I think out of all the options proposed, 'identity' is the most readable/quickly understandable.

For example, I think the use of 'identity' reads very nicely in `[1,2,3,4].group_by(&:identity)`

Except that #identity seems to imply the same thing as #\_\_id\_\_, and "a"#\_\_id\_\_ is not necessarily == "a"#\_\_id\_\_, as Matz said earlier.

The advantage of using #reference is that there's no existing method or concept we're overloading; there are no such things as "reference" objects (or "pointers") in Ruby -- it is understood that all references are automatically dereferenced when operated on -- so returning the references and then comparing them should be understood by most rubyists as effectively the same as comparing the objects directly (whatever "directly" means).

tl;dr:

Defining #identity that conflicts with #\_\_id\_\_ is confusing, however slightly it might be.

**#38 - 06/25/2013 07:24 AM - headius (Charles Nutter)**

phluid61 (Matthew Kerwin) wrote:

charliesome (Charlie Somerville) wrote:

I think out of all the options proposed, 'identity' is the most readable/quickly understandable.

For example, I think the use of 'identity' reads very nicely in `[1,2,3,4].group_by(&:identity)`

Except that #identity seems to imply the same thing as #\_\_id\_\_, and "a"#\_\_id\_\_ is not necessarily == "a"#\_\_id\_\_, as Matz said earlier.

I have this concern as well. Having "a"#\_\_id\_\_/object\_id and "a".identity return drastically different things feels like it will just be confusing.

**#39 - 06/25/2013 07:46 AM - vjoel (Joel VanderWerf)**

Another argument against #identity: it is used by several libraries for something completely different. For example, in narray:

```
>> NMatrix.float(2,2).identity
=> NMatrixfloat2,2:
[ [ 1.0, 0.0 ],
  [ 0.0, 1.0 ] ]
```

It's also used in celluloid-zmq:

```
s1 = PubSocket.new
s1.identity = "publisher-A"
```

I vote for "itself" or "self", which are unlikely to be defined anywhere with some fundamentally different meaning.

**#40 - 08/31/2013 06:59 AM - marcandre (Marc-Andre Lafortune)**

- *File self.pdf added*

Slide attached.

I hope to win the prize for simplest slide too.

**#41 - 08/31/2013 12:14 PM - charliesome (Charlie Somerville)**

marcandre: I think you made a mistake in your slide. It says "Returns the class of obj", but it should say "Returns obj"

**#42 - 08/31/2013 01:25 PM - matz (Yukihiro Matsumoto)**

I can accept #itself. I want to see it isn't conflict with existing methods.

Matz.



**#43 - 01/20/2014 12:48 AM - avit (Andrew Vit)**

Rails ActiveSupport includes a similar method called presence. There is also a request to add block support to it, for a similar purpose: <https://github.com/rails/rails/pull/13416#issuecomment-32636227>

**#44 - 01/20/2014 05:47 AM - marcandre (Marc-Andre Lafortune)**

Andrew Vit wrote:

Rails ActiveSupport includes a similar method called presence.

Mmm, no, that's quite different. `"".presence # => nil` for example.

**#45 - 01/20/2014 02:30 PM - fuadksd (Fuad Saud)**

Wouldn't such method accepting a block remove the need to have `Object#tap` at all? As I understand this method is just a tap that doesn't need a block.

--

Fuad Saud  
Sent with Airmail

**#46 - 01/20/2014 05:16 PM - sawa (Tsuyoshi Sawada)**

I would like to propose receiver as the method name.

**#47 - 01/20/2014 08:20 PM - phluid61 (Matthew Kerwin)**

On Jan 21, 2014 12:29 AM, "Fuad Saud" [fuadksd@gmail.com](mailto:fuadksd@gmail.com) wrote:

Wouldn't such method accepting a block remove the need to have `Object#tap` at all? As I understand this method is just a tap that doesn't need a block.

That depends on the contract. I was under the impression that `#itself` (or whatever name) in block form would return the value of the block. e.g:

```
def tap
  yield self if block_given?
  self
end
def itself
  if block_given?
    yield self
  else
    self
  end
end
```

**#48 - 01/20/2014 08:42 PM - fuadksd (Fuad Saud)**

That is interesting behaviour for chaining; not sure if consistent though.

—  
Sent from Mailbox for iPhone

**#49 - 01/24/2014 09:19 PM - avit (Andrew Vit)**

Mmm, no, that's quite different. `"".presence # => nil` for example.

Yes, I was aware of the differences from ActiveSupport presence, I just wanted to point out the similar need for chaining "itself" with a block.

The proposed "itself" method also looks similar to "Enumerable#map" but for a single object, effectively:

```
["ruby"].map {|n| n.upcase }.first
# same as:
"ruby".itself {|n| n.upcase }
```

But I suppose "map" as a method name would be out of the question.

Has anyone considered "yield" as the method name? It also seems to fit well, and would likely not conflict with anything:

```
class Object
  def yield
    if block_given?
      yield self
    else
      self
    end
  end
end
```

```
# yield as a noun means "the result": an object's result (its yield) is itself
"ruby".yield #=> "ruby"
```

```
# yield as a verb means "give way to" or "produce": the object gives way to the block
"ruby".yield {|s| s.upcase } #=> "RUBY"
```

Although there might be confusion between yield as a keyword and self.yield as a method, I do like this symmetry:

```
yield self          # calls block with self, returns result
self.yield(&block) # calls block with self, returns result
self.yield         # implicit identity, just like: {|obj| obj }
```

#### #50 - 07/28/2014 04:25 AM - nobu (Nobuyoshi Nakada)

- Description updated

Yukihiro Matsumoto wrote:

I can accept #itself. I want to see it isn't conflict with existing methods.

It's hard to imagine other functionality from that name for me.  
The best way to tell if it will break something should be implementing it before previews, IMHO.

#### #51 - 07/28/2014 04:37 AM - duerst (Martin Dürst)

- Status changed from Feedback to Open

#### #52 - 08/01/2014 05:18 PM - rafaelfranca (Rafael França)

We just implemented it on Active Support <https://github.com/rails/rails/commit/702ad710b57bef45b081ebf42e6fa70820fdd810>

I believe matz already accepted it as #itself so we are renaming it to #itself.

#### #53 - 08/01/2014 06:23 PM - rafaelfranca (Rafael França)

- File itself.diff added

Here the implementation. This is my first patch to Ruby so I'm not sure it is correct. Let me know if I need to change something.

#### #54 - 08/02/2014 01:27 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

Applied in changeset r47028.

object.c: Object#itself

- object.c (rb\_obj\_itself): new method Object#itself. based on the patch by Rafael França in [ruby-core:64156]. [EXPERIMENTAL] this method may be renamed due to compatibilities. [ruby-core:44704] [Feature [#6373](#)]

#### #55 - 08/20/2015 07:02 AM - laise (Alexey Chernenkov)

-1

For me it would be better to accept #47 - it improves good old #tap and introduces a new #itself with block.

Copied here:

```
def tap
  yield self if block_given?
```

```
  self
end

def itself
  if block_given?
    yield self
  else
    self
  end
end
end
```

## Files

---

self.pdf	77.8 KB	08/31/2013	marcandre (Marc-Andre Lafortune)
itself.diff	1.87 KB	08/01/2014	rafaelfranca (Rafael França)