Ruby - Misc #19304

Kernel vs Object documentation

01/03/2023 02:00 PM - zverok (Victor Shepelev)

Status:	Assigned	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	

Description

The restating of the problems raised in #19300, from scratch.

I believe it is rather a topic of **community significance** and would be happy if it is possible to **discuss on dev-meeting**, even if the outcome would probably "let's change RDoc in this or that way".

So, the problem statement:

1. Kernel module defines two "types" of methods: private ones, that pretend to be global (like puts), but actually available from inside every object; and public ones (like #class or #frozen?) that are present in every object including Kernel.
2. Since the beginning of times, docs provided an illusion that the second type of the methods belongs to Object class, which is, in fact, devoid of its own definitions, just includes Kernel. This was handled by special RDoc hack (which, basically, forcefully reattached definition of public methods if they are defined in Kernel, to Object)
3. The RDoc hack was working in C code only, so after introduction of kernel.rb the illusion started to crumble: methods like #tap, #then, #class are now documented as belonging to Kernel (breaking the tradition of public methods being documented in Object)
4. This is all inconsistent and confusing, and I believe, adds to friction both for newcomers and curious investigators of the language!

Additionally, I believe:

1. That the distinction of "two kinds of methods" is useful (saying from a practical experience with explaining Ruby while mentoring, writing articles, and discussing with colleagues)
2. But, considering everything, I agree with what @Eregon (Benoit Daloze) and @nobu (Nobuyoshi Nakada) say in #19300: pretending that methods belong not to the module they really belong to is not the optimal way to document things!

So, the options I see (save for "do nothing and let things sort themselves somehow"):

1. Change Ruby's implementation so public methods would really belong to Object. Actually, I don't believe anybody would agree to experiment on this scale, so just listing it here for completeness.
2. Just remove the RDoc hack; all methods would belong to Kernel, and maybe some introductory free-form text will instruct that they are different. **Pro:** Easy to do. **Contra:** The Kernel docs would be super-confusing.
3. Continue to maintain the hack in RDoc and extend it to kernel.rb (so methods like #clone and #frozen? would be still in Object). **Pro:** Relatively easy to do, will maintain structure many people used to. **Contra:** This is still a lie, methods belong to Kernel, not Object.
4. Adjust RDoc to a) be able to separately list **public** and **private** methods in two different lists, and add intro for Kernel explaining how to treat those. **Pro:** Probably the best correspondence to reality? **Contra:** Not sure how easy it is to change RDoc this way; and other renderers (like ruby-doc.com and rubyapi.org) would need to adjust themselves.

So far, (4) looks the most reasonable (if (1) is 100% impossible, that is!)

History

#1 - 01/03/2023 04:35 PM - austin (Austin Ziegler)

I think that (2) looks promising if sections are used (https://ruby.github.io/rdoc/RDoc/Markup.html#class-RDoc::Markup-label-Sections) in the documentation, although I must admit that I've never used those. It would absolutely need to be used consistently in both C and kernel.rb method definitions.

#2 - 01/04/2023 06:27 AM - nobu (Nobuyoshi Nakada)

Is the option 3 like this?
https://github.com/ruby/rdoc/pull/961

#3 - 01/04/2023 08:21 AM - sawa (Tsuyoshi Sawada)

Can you expand on why option 2 would be super-confusing? I think that option is the best.

#4 - 01/04/2023 12:14 PM - zverok (Victor Shepelev)

@sawa

> Can you expand on why option 2 would be super-confusing? I think that option is the best.

Because those are two different categories of methods (as this and the previous ticket discuss in detail), and having them in one mixed list makes it hard to distinguish "global methods" (aka Kernel's private) from "public methods that every object has".

From every Ruby documentation user, it doesn't matter much that they are defined in the same module, but it *does* matter much that methods like puts or require aren't homogenous with methods like class, frozen? or clone.

That, I believe, was the reason for the initial RDoc decision to pretend public methods are belonging to Object.

#5 - 01/05/2023 12:49 AM - sawa (Tsuyoshi Sawada)

@zverok (Victor Shepelev) The Kernel page is entirely broken anyway from its first sentence:

> The Kernel module is included by class Object, so its methods are available in every Ruby object.

the latter half of which is not true since the introduction of BasicObject, and what it claims to be "Public Instance Methods" are not public. It is confusing in the current state.

It just needs to be entirely fixed. Fixing this and describing public and private methods correctly would not be confusing.

#6 - 01/05/2023 07:34 AM - zverok (Victor Shepelev)

> Fixing this and **describing public and private methods correctly** would not be confusing.

This is option (4) of my initial list, I believe.

Option (2) — the easiest and the least desirable of them — is just to dump everything into Kernel without trying to separate the two kinds.

(If you meant only fixing each individual method docs, without separating methods in two lists, it will still be bad. The list of methods that class/object has is a significant part of the docs navigation. Having puts and class in the same homogenous list is bad.)

#7 - 01/17/2023 07:58 PM - matheusrich (Matheus Richard)

Thanks for voicing that, @zverok (Victor Shepelev)! I always found this confusing (i.e., some methods being defined in Kernel, not in Object), and I'd love to see the docs reflect that in some form. (1) feels like the right thing to do, so what are the roadblocks? Backward compatibility? After that, I think (4) is the way to go.

#8 - 02/10/2023 01:28 AM - mame (Yusuke Endoh)

Discussed at the dev meeting. In conclusion, Option 4 looks good.

- Stop that "hack" of RDoc; all methods defined in Kernel should be listed in the Kernel documentation.
- rdoc should clearly distinguish between module functions and others. (This is not limited to Kernel)
- It is okay to state, at the beginning of the Kernel documentation, that Kernel's module functions are a global function in effect.

The current RDoc does not appear to handle module functions correctly. For example, Math.acos is listed as "Public Class Methods", which would not be appropriate. If this is fixed, the problem of this ticket should be also fixed automatically, we thought.

For the record, matz stated that he does not feel the need to distinguish between global functions (e.g., fork and exit) and others (e.g., is_a? and kind_of?), either in the implementation or in the documentation. He is fine with all methods being presented in one list. But he was not opposed to the above handling.

#9 - 04/03/2024 03:50 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned