Ruby - Misc #18803

DevMeeting-2022-06-16

05/25/2022 03:59 AM - mame (Yusuke Endoh)

Status:	Closed	
Priority:	Normal	
Assignee:		

Description

The next dev meeting

Date: 2022/06/16 13:00-17:00 (JST)
Log: https://github.com/ruby/dev-meeting-log/blob/master/DevMeeting-2022-06-16.md

- Dev meeting *IS NOT* a decision-making place. All decisions should be done at the bug tracker.
- Dev meeting is a place we can ask Matz, nobu, nurse and other developers directly.
- Matz is a very busy person. Take this opportunity to ask him. If you can not attend, other attendees can ask instead of you (if attendees can understand your issue).
- We will write a record of the discussion in the file or to each ticket in English.
- All activities are best-effort (keep in mind that most of us are volunteer developers).
- The date, time and place of the meeting are scheduled according to when/where we can reserve Matz's time.
- *DO NOT* discuss then on this ticket, please.

Call for agenda items

If you have a ticket that you want matz and committers to discuss, please post it into this ticket in the following format:

```
* [Ticket ref] Ticket title (your name)
  * Comment (A summary of the ticket, why you put this ticket here, what point should be discussed
, etc.)
```

Example:

```
* [Feature #14609] `Kernel#p` without args shows the receiver (ko1)
  * I feel this feature is very useful and some people say :+1: so let discuss this feature.
```

- It is recommended to add a comment by 2022/06/13. We hold a preparatory meeting to create an agenda a few days before the dev-meeting.
- The format is strict. We'll use this script to automatically create an markdown-style agenda. We may ignore a comment that does not follow the format.
- Your comment is mandatory. We cannot read all discussion of the ticket in a limited time. We appreciate it if you could write a short summary and update from a previous discussion.

Related issues:

Related to Ruby - Misc #14770: [META] DevelopersMeeting	**Open**

History

#1 - 05/25/2022 03:59 AM - mame (Yusuke Endoh)

- Related to Misc #14770: [META] DevelopersMeeting added

#2 - 06/01/2022 02:01 AM - shugo (Shugo Maeda)

- [Bug #18806] protected methods defined by refinements can't be called (shugo)
  - I prefer "1. Treat defined methods as though they were defined on the refined class. "
  - The pull request looks fine: https://github.com/ruby/ruby/pull/5966

#3 - 06/02/2022 09:35 AM - byroot (Jean Boussier)

- [Bug #18813] Module#autoload isn't strict about the autoloaded constant (fxn)
  - module M; autoload :OpenSSL, "openssl"; end works but is inconsistent.
  - M.constants(false) # => [:OpenSSL]
  - M::OpenSSL # => ::OpenSSL

- M.constants(false) # => []
- Should we break it? deprecate it? do nothing?
- There are some important gems relying on this currently, including bundler.

#4 - 06/02/2022 01:35 PM - kddnewton (Kevin Newton)

- [Feature #18773] Pass an optional range object to deconstruct
    - It can be very expensive to compute the array for matching against deconstruct
    - By passing a range object we can quickly dismiss matches that won't work
    - This can be done in a backward-compatible way by checking the arity of deconstruct

#5 - 06/06/2022 06:17 PM - jeremyevans0 (Jeremy Evans)

- [Feature #18788] Support passing Regexp options as String to Regexp.new (jeremyevans0)
    - Do we want to add support for Regexp.new(code, options), where options is a string (e.g. 'im').
    - @nobu (Nobuyoshi Nakada) pointed out it is already possible to get identical behavior using Regexp.new("(?#{options}:#{code})").
- [Feature #18749] Strangeness of endless inclusive ranges (jeremyevans0)
    - Currently, we support both endless inclusive ranges and endless exclusive ranges.
    - Endless exclusive ranges do not contain endless inclusive ranges.
    - Do we want to convert endless inclusive ranges to endless exclusive ranges?
    - Do we want to keep supporting both, but consider them equal?
- [Feature #18461] closures are capturing unused variables (jeremyevans0)
    - Should procs capture local variables they do not use syntactically inside the proc?
    - Currently, they do, which I believe is expected and desired.
    - Removing this ability would remove the ability to using eval inside the proc to get access to the local variable.
    - Changing the behavior could allow for earlier garbage collection, and potentially improve performance.
- [Feature #18279] ENV.merge! support multiple arguments as Hash.merge! (jeremyevans0)
    - Can we add this ability?
    - @nobu (Nobuyoshi Nakada) has already prepared a patch.

#6 - 06/14/2022 02:31 AM - hsbt (Hiroshi SHIBATA)

- [Feature #18159] Integrate functionality of dead_end gem into Ruby (@duerst (Martin Dürst))
    - How about this?
    - IMO: We try to add dead_end as the default gems before releasing 3.2.0-preview2

#7 - 06/14/2022 10:40 AM - Eregon (Benoit Daloze)

- [Bug #18729] Method#owner and UnboundMethod#owner are incorrect after using Module#public/protected/private (eregon)
    - Please see https://bugs.ruby-lang.org/issues/18435#note-12
    - Removing "ZSUPER methods" solves everything: simpler semantics, easier to understand for everyone and Method/UnboundMethod objects behave like people expect, no special edge case for "ZSUPER methods". Also solves #18729 in a very simple way. And it is already what JRuby & TruffleRuby do.
    - "ZSUPER methods" seems accidental complexity from long ago, I believe it is time to remove it.
- [Bug #18826] Symbol#to_proc inconsistent, sometimes calls private methods (eregon)
    - Let's fix so it never calls private methods at least
    - What about protected methods? I think it would be best for Symbol#to_proc procs to only call public methods, never protected/private, otherwise we have deep inconsistencies and complexity.

#8 - 06/17/2022 09:43 AM - mame (Yusuke Endoh)

- Description updated

- Status changed from Open to Closed