The next dev meeting

Date: 2020/06/18 13:00-17:00
Place/Agenda/Log: [https://github.com/ruby/dev-meeting-log/blob/master/DevelopersMeeting20200618Japan.md](https://github.com/ruby/dev-meeting-log/blob/master/DevelopersMeeting20200618Japan.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 log about the discussion to a 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 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.

Comment deadline: 2020/06/11 (one week before the meeting)
- The format is strict. We'll use [this script to automatically create an markdown-style agenda](https://github.com/ruby-dev-meeting-log/dev-meeting-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.

Related issues:

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

**History**

#1 - 06/04/2020 01:25 AM - mame (Yusuke Endoh)
- Related to Misc #14770: [META] DevelopersMeeting added

#2 - 06/04/2020 01:32 AM - mame (Yusuke Endoh)
- Description updated

#3 - 06/04/2020 03:37 AM - hsbt (Hiroshi SHIBATA)
* [Bug #8446] sdbm fails to fetch existing key if many elements in it
  * sdbm library is not maintain while a long time. Should we remove it from ruby repo skipping the bundled gems?
  * [https://bugs.ruby-lang.org/issues/8446#note-5](https://bugs.ruby-lang.org/issues/8446#note-5)
* [Feature #16963] Remove English.rb from Ruby 2.8/3.0
  * Should we remove English.rb?

#4 - 06/04/2020 03:39 PM - jeremyevans0 (Jeremy Evans)
● [Bug #9573] descendants of a module don't gain its future ancestors, but descendants of a class, do (jeremyevans0)
  ○ Support for Module#include was implemented 3 months ago
  ○ Bugs prevented support for Module#prepend at the time
  ○ I've since fixed all bugs
  ○ Implementing support for Module#prepend is now straightforward and causes no test failures, is it OK to merge the patch?

● [Feature #14267] Lazy proc allocation introduced in #14045 creates regression (jeremyevans0)
  ○ I added a patch implementing Proc#=== and #eql? (previously not implemented, so Object#=== and #eql? were used)
  ○ Do the equivalence conditions I describe make sense, or should more be added?
  ○ Is it OK to merge the patch?

● [Bug #11698] inconsistent behavior of refining frozen class (jeremyevans0)
  ○ Refining a frozen class where the refinement adds a method is currently broken.
  ○ Refinement methods are internal, so I'm not sure we should raise FrozenError in this case
  ○ Is it OK to merge the patch?

● [Bug #16504] foo(*args, &args.pop) should pass all elements of args (jeremyevans0)
  ○ Bug is due to an optimization that skips duping of args for the splat.
  ○ I created a patch that fixes the issue in most cases by removing the optimization in those cases.
  ○ The patch does not remove the optimization in common cases, such as &local_variable or &:symbol
  ○ Truly fixing the issue in all cases requires removing the optimization completely.
  ○ Is it OK to merge the patch?

● [Feature #16470] Issue with nanoseconds in Time#inspect (jeremyevans0)
  ○ Currently, Time#inspect can display fractional seconds as a rational, which few rational people want.
  ○ I think Time#inspect should always display fractional seconds as decimal.
  ○ I have a patch to do this, but it makes Time#inspect the same for Time instances that are not equal.
  ○ Arbitrary precision for fractional seconds, the benefit of storing nanoseconds as a rational, seems pointless to me.
  ○ Do we want to keep things as is, merge the patch, or change Time's implementation so fractional seconds are not stored with greater than nanosecond precision?

#5 - 06/06/2020 06:01 PM - Eregon (Benoit Daloze)

● [Feature #16378] Support leading arguments together with ... (eregon)
  ○ Already accepted for master, could someone review the PR from Jeremy?
  ○ Could we backport it to 2.7.2? So then there is a non-ruby2_keywords and long-term way to do delegation for more cases.

#6 - 06/06/2020 06:04 PM - Eregon (Benoit Daloze)

● [Feature #16897] General purpose memoizer in Ruby 3 with Ruby 2 performance (eregon)
  ○ Thoughts about the ***args / Arguments proposal?
  ○ Ideas to optimize delegation with *args, **kwargs better, which it seems most people expect (it's intuitive, same as Python) is the Ruby 3 way to do delegation?
  ○ Some other ideas to optimize memoization, maybe using ...?

#7 - 06/08/2020 06:49 PM - dsisnero (Dominic Sisneros)

● [Feature #14722] python's buffer protocol clone (dsisnero)
  ○ many C-extensions that use large buffer like objects in C - Numo::Narray, NMatrix, red-arrow, but to transfer between them and ruby
    usually copy the data to go between types
  ○ create a c-api to describe the shape and access of the data to avoid copying data
  ○ maybe use the arrow c data interface https://arrow.apache.org/docs/format/CDataInterface.html

#8 - 06/08/2020 07:43 PM - dsisnero (Dominic Sisneros)

● [Feature #12901] Anonymous functions without scope lookup overhead. (dsisnero)
  ○ this allows performance bump
  ○ also can allow to serialize functions - #11630 or (isolated procs)
  ○ can add multi-method variant if adding new syntax

      func add(x,y){ x + y }
      mfunc add(x: Int, y: Int){ __intrinsic__.int_add(x,y)
      mfunc add(x: String, y: String){ __instinsic__.string_add(x,y) }

#9 - 06/11/2020 02:17 AM - jeremyevans0 (Jeremy Evans)

● [Bug #14541] Class variables have broken semantics, let's fix them (jeremyevans0)
- My previous commit to fix this only raised in verbose mode, since the class variable overtaken warning was only issued in verbose mode.
- Are we OK jumping directly from verbose-mode warning in 2.7 to RuntimeError in 3.0, or do we want to issue a warning even in non-verbose mode in 3.0 and switch to RuntimeError in 3.1?

#10 - 06/11/2020 07:31 AM - mame (Yusuke Endoh)
- [Feature #16950] Stop nonsense keyword argument warnings in 2.6 (mame)
  - 2.6 produces a warning that no longer makes sense. Unfortunately, this warning is not only unhelpful but also annoying for supporting 3.0 keyword behavior (according to some Rails developers). This is a change for 2.6, but how about deleting the warning?

#11 - 06/12/2020 08:11 PM - connorshea (Connor Shea)
- [Feature #16456] Ruby 2.7 argument delegation (...) should be its own kind of parameter in Method#parameters (aaronc81)
  - Right now, the parameters method returns [[rest, :*], [block, :&]] if a method uses the ... syntax.
  - It'd be nice if the returned array were more clear about the method using ....
  - Can we change the returned values to something like [:rest, :"..."], [:block, :"..."], especially now that ... can be used after other parameters?

#12 - 06/14/2020 06:03 AM - k0kubun (Takashi Kokubun)
- [Misc #16961] Is overriding a method in a subclass considered as a breaking change or not? (k0kubun)
  - Is it acceptable to override Integer#zero? which is currently handled by Numeric#zero? to optimize it on MJIT?
  - How should we make this kind of decision? (the main topic of the ticket)

#13 - 06/14/2020 11:11 AM - zverok (Victor Shepelev)
- [Feature #15822] Add Hash#except (zverok)
  - Last time it was discussed, Matz asked "We didn't see the need for Hash#except yet. Any (real world use-case)? I don't think the name except is the best name for the behavior." In the ticket's new comments, I am providing use cases and name justification.
  - I vaguely remember already adding it to some previous meetings agenda, but I don't see any outcome in the ticket

#14 - 06/16/2020 10:07 AM - byroot (Jean Boussier)
- [Feature #16848] Allow callables in $LOAD_PATH (byroot)
  - I added a proposal for the "loader" interface based on two methods: find_feature and load_feature.
  - A couple solutions for the $LOAD_PATH backward compatibility concern were proposed (maintaining a distinct $FEATURE_LOADERS array).
  - I'd like to really question the importance of that compatibility issue. It would certainly require some code to be updated, but wouldn't break any use case.
  - More generally I'd like to know what I could do to help move this forward.

#15 - 06/18/2020 02:09 AM - mrkn (Kenta Murata)
- [Feature #16812] Allow slicing arrays with ArithmeticSequence
  - I found the inconsistency between the existing behaviors: [*0..10][-100..100] vs [*0..10][..100].
  - I believe it is better to fix the behavior of [*0..10][-100..100].

#16 - 06/19/2020 08:04 AM - mame (Yusuke Endoh)
- Status changed from Open to Closed