The next dev meeting

Date: 2022/08/18 13:00-17:00 (JST)
Log: TBD

- 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 2022/08/15. 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 master - Misc #14770: [META] DevelopersMeeting Open

History

#1 - 08/04/2022 04:37 PM - mame (Yusuke Endoh)
- Related to Misc #14770: [META] DevelopersMeeting added

#2 - 08/06/2022 01:37 PM - zverok (Victor Shepelev)

- [Feature #18368] Range#step semantics for non-Numeric ranges (zverok)
  - Proposal to reimplement (a..b).step(c) via +, or provide another method with this behavior
  - More clarifications provided & answers to the questions after last dev.meeting
- [Bug #18580] Range#include? inconsistency for beginless String ranges
  - "This was discussed during the February 2022 developer meeting, and @matz (Yukihiro Matsumoto) said he needs more time to consider it."
- [Feature #17330] Object#non (zverok)
  - Proposal to introduce generic "nullify if unacceptable" API (zverok)
  - more explanations and examples provided
- [Feature #16122] Struct::Value: simple immutable value object (zverok)
  - The ticket is in weird state. It has "Status: Feedback", but generic state "CLOSED", but I don't see any notice of HOW it was
closed?.. Is it RedMine glitch?.. I don't see an option to reopen it.

- The latest comment from Matz seems to lean towards accepting proposal, the only culprit is a name: "Struct::Value seems acceptable to him, but he wanted to seek a better name. Devs suggested Tuple, NamedTuple, and Record, but none of them seemed to fit for him."
- Can we somehow move forward with this?
- [Feature #18934] Introduce method results memoization API in the core (zverok)
- New ticket, proposing and justifying simple memoized def method_without_args

#3 - 08/08/2022 07:20 AM - byroot (Jean Boussier)

- [Feature #18944] Add SizedQueue#push(timeout:) (byroot)
  - Matz said this should be required separately.
  - When implementing Queue#pop(timeout:) it was very evident this was needed.
  - Spec is return self on success, nil on timeout, like pop.
  - Useful to periodically do heartbeat, drop messages, etc when waiting to push on a queue.
- [Feature #18885] End of boot advisory API (byroot)
  - e.g. something like RubyVM.prepare, or RubyVM.ready.
  - Would allow applications to notify the VM that they're done loading and about to process user input.
  - Could be used by the VM to enable some optimizations, such as precomputing constant caches, etc.
  - Particularly useful for forking server, but not only.

#4 - 08/09/2022 07:26 PM - jeremyevans0 (Jeremy Evans)

- [Bug #18955] Kernel#sprintf - %c ignores a non-ASCII character's encoding (jeremyevans0)
  - Do we want the %c format specifier to transpose a string argument to the format string's encoding before taking the first character?
- [Bug #18958] Kernel#sprintf doesn't apply format sequence in some encodings (jeremyevans0)
  - Should we raise an ArgumentError when the format string is in an incompatible encoding?
  - The encodings listed in the issue are not the same as those that are not ascii_compatible?.
  - If we do want to raise an exception for this, do we want a new encoding flag for this, or a hardcoded list of incompatible encodings?
- [Bug #18797] Third argument to Regexp.new is a bit broken (jeremyevans0)
  - Do we want to deprecate passing a third argument to Regexp.new?
- [Bug #18767] IO.foreach hangs up when passes limit=0 (jeremyevans0)
  - Is the current infinite loop in this case expected, or should it raise an ArgumentError as IO.readlines does?

#5 - 08/10/2022 10:30 AM - Eregon (Benoit Daloze)

- [Feature #18949] Deprecate and remove replicate and dummy encodings (eregon)
  - I will try to attend the dev meeting. Would it be possible to discuss this towards the end of the meeting, 16 JST (= 9am CEST) my preference, or if better 15 JST (= 8am CEST)?
  - There are multiple things to decide, see https://bugs.ruby-lang.org/issues/18949#To-Decide
    1. (Encoding#replicate and rb_define_dummy_encoding()) is most important (for complexity & performance) and given the extremely low usage it seems safe to deprecate and then remove
    1. Handling the dummy UTF-16 and UTF-32 encodings, seems used very rarely but harder to assess. It causes significant performance overhead. Discussion e.g. in https://bugs.ruby-lang.org/issues/18949#note-23
    1. Multiple devs want to keep (other) existing dummy encodings based on usages, so let's keep them for now. Maybe we could make them non-dummy or only available for transcoding in the future.

#6 - 08/12/2022 05:24 PM - palkan (Vladimir Dementyev)

- [Feature #18408] Allow pattern match to set non-local variables
  - That would make, in particular, 42 => @a possible
    - This addition seems natural after we added non-local vars support for pinning
    - It also "restores" the original rightward assignment functionality
  - Here is a PR attached: https://github.com/ruby/ruby/pull/5426
- [Feature #18821] Expose Pattern Matching interfaces in core classes
  - Here is a PR adding MatchData#[deconstruct,deconstruct_keys] waiting for feedback: https://github.com/ruby/ruby/pull/6216

#7 - 08/15/2022 11:08 AM - Eregon (Benoit Daloze)

- [Bug #18726] Method#owner and UnboundMethod#owner are incorrect after using Module#public/protected/private (eregon)
  - Also [Bug #18436] and [Bug #18751], they are all the same issue basically.
  - Let's attempt to stop hiding ZSUPER methods, this was never tried and I think would cause little incompatibilities. OK to try this?
The semantic model after that change will finally make sense for users and Ruby implementers, instance_methods reflects the method table, owner means in which module's method table the method is. Also this will mean ZSUPER methods are just an internal implementation detail on CRuby, other Ruby impls are free to use something else like shallow-copy (like alias_method does) but instance_methods/owner/etc will be the same for the user. The hiding is actually "exposing" ZSUPER methods to Ruby users because other Ruby implementations don't use ZSUPER methods and don't hide the visibility-created methods.

#8 - 08/18/2022 07:14 AM - hsbt (Hiroshi SHIBATA)

- [Feature #18159] Integrate functionality of syntax_suggest gem into Ruby
  - Is it ok for syntax_suggest?

#9 - 08/18/2022 09:37 AM - mame (Yusuke Endoh)

We could not finish all the agenda in time. We will continue on 25th.

#10 - 08/20/2022 07:51 PM - Eregon (Benoit Daloze)

- [Feature #18798] UnboundMethod#== with inherited classes (eregon)
  - OK to change UnboundMethod#== to check if same method definition (and ignore from which class instance_method was used on)?
  - If not, OK to add {Method,UnboundMethod}#same_definition?(other)?