DevelopersMeeting20191017Japan

https://bugs.ruby-lang.org/issues/16232

Venue

- 10/17 (Thu) 13:00-17:00 @ pixiv Inc. (**6F**)
    - up to 6F, and ask someone to call usa-san in Studio

Attendees

Add your name (or ask a committer to invite you)

- matz (remote)
- mame (remote)
- usa (venue staff)
- znz (remote)
- nobu (remote)
- aycabta
- osyo (invited by aycabta)
- a_matsuda
- mrkn (remote)
- akr
- Soutaro
- knu

Absent:

- Martin Dürst (trip to Unicode Conference)
- Shyouhei (is sick)

Next Date

- 11/12 (Tue) 13:00-17:00 @ full remote

    - Review all 2.7 new features
- 11/28 (Thu) 13:00-17:00 @ pixiv 6F studio
    - Monthly meeting; it would be the last change for 2.7 new features

Announce

About 2.7 timeframe

- preview 2
    - naruse: within a few days, maybe

Next Stable releases timeframe

- usa: maybe the 2nd week of December?
- mame: need to ask nagachika-san

Check security tickets

Topics

[Feature #16029] Expose fstring related APIs to C-extensions (byroot)

- Apparently the current implementation is complicated to expose, but no details were provided.
- Is there something I or others could change to the implementation so that it could be exposed?

Discussion:

- nobu: does it make speed up?
- naruse: yes see also https://twitter.com/search?q=from%3Afrsyuki until%3A2017-05-04&src=typed_query
- mame: will the strings be GC-ed?
- nobu: yes
- ko1: it's a little dangerous
- nobu: the dangerousness is not of fstring but of shared string

- ko1: exposing these APIs may introduce many bugs on corner cases, I guess

[Feature #16038] Provide a public WeakMap that compares by equality rather than by identity (byroot)

- Matz asked for real world use-case, which I did provide.
- In short it's useful for implementing value objects deduplication, just like how the fstring table works, but for user defined, more complex types.

Discussion:

- mame: the use-case is for memorize
- usa: it's reasonable
- ko1: is GC-scheduled objects still be visible, isn't it?
- nobu: lazy sweep checks it
- ko1: is finalizer OK? … ah, it may be OK
- mame: is it OK that the referenced object is lazy sweeping?
- nobu & matz: there is no such case by design
- ko1: is compaction safe? only Aaron knows :)
- mame: the proposal requests an interface to use equality based map
- mame: introduce a new keyword argument for the flag?
- matz: I don't disagree the feature itself
- ko1: for implementation, we have to make a new compare function for st?
- nobu: maybe
- ko1: and make a new check for the existance…
- nobu: maybe
- akr: I don't like the API, compare_by_equality, because it introduces extra complexity: When it is called with non-empty WeakMap, it may remove some elements.
- mame: I agree akr. there might be much better API
- akr: I think that this proposal expects an ideal GC. The real application may behave badly if GC is invoked too frequently or too rare.
- usa: at all, should we implement this API in 2.7 or not?
- matz: we don't have to be rough-and-ready in this case

[Bug #10314] Default argument lookup fails in Ruby 2.2 for circular shadowed variable names (jeremyevans)

- Do we want to change `def foo(bar=bar)` from warning to SyntaxError using the patch?

Discussion:

- nobu: we can make this an error now
- matz: go ahead
- ko1: method body?
- mame: should be in another ticket. should warn before to be an error
- ko1: matz, is the change ok?
- matz: ok

[Bug #11055] autoload resets private_constant (jeremyevans)

- Do we want to copy constant visibility information across the autoload using the patch?

Discussion:

- usa: is this bug?
- matz: maybe. but if a library does not recognize the constant is private, how autoload behave?
- nobu?: Current implementation is to delete the constant from the entry, and add new one, then private flag is cleared
- matz: how to fix?
- nobu: I'll look this in a few days.

[Bug #13249] Access modifiers don't have an effect inside class methods in Ruby >= 2.3 (jeremyevans)

- Do we want to add a warning for misuse of method visibility methods using the patch?

Discussion:

- already fixed.

[Bug #15267] File.basename + File.extname does not restore the original name (jeremyevans)

- Do we want to make them restore the original name using the patch?

Discussion:

```
name = 'file.'

File.basename(name, '.*') #=> "name"

File.extname(name) #=> ""
```

- usa: should fix `File.extname`, not `File.basename`, because of their usecases, if we do
- `File.extname("foo.")` will be `"."`
- matz: ok

[Feature #16245] Add interfaces to count and measure size all IMEMO objects (Sam Saffron)

- Would love to see this included in 2.7, refined proposal does not add any new APIs.

Discussion:

```
RubyVM.stat

{

   :global_method_state=>143,

   :global_constant_state=>1369,

   :class_serial=>8768,

   :imemo_ment_count,

   :imemo_iseq_count,

   :imemo_env_count,

   :imemo_tmpbuf_count,

   :imemo_ast_count,
```

```
    :imemo_ment_size,
    :imemo_iseq_size,
    :imemo_env_size,
    :imemo_tmpbuf_size,
    :imemo_ast_size
}
```

- ko1: I've already responded.

[Feature #13683] Add strict Enumerable#single (rafaelfranca)

- Matz didn't like the single name but we have a proposal for #only

Discussion:

```
[1].only    #=> 1
[1, 2].only #=> ArgumentError
```

- ko1: pattern match may be helpful for this use case?

```
case [1]    ; in (x); p x; end #=> [1]
case [1, 2]; in (x); p x; end #=> [1, 2]
case [1]    ; in [x]; p x; end #=> 1
case [1, 2]; in [x]; p x; end #=> NoMatchingPatternError
```

- a_matsuda: This is actually wanted in Rails applications. It named "#pick".
- matz: I hate #only. #first! is not good because it does not represent the behavior
- naruse: the name need to include both checking

```
case [1, 2]
```

```
in [x]

  p x

end

#=> t.rb:3:in `<main>': [1, 2] (NoMatchingPatternError)
```

[Bug #16143] BOM UTF-8 is not removed after rewind (kou)

- Can we commit this? I've reviewed. I think that this is ready to merge.

Discussion:

- usa: `rewind` doesn't sound like encoding aware.
  - `IO#rewind` is really low level operation, we don't call it usually.
- `CSV#rewind` should handle BOM itself, not `IO#rewind`.
- Should provide a method to know BOM is read?
- Adding an option `io.rewind(respect_bom: true)` can be an option???
  - `rewind(to_after_bom: true)`
  - `rewind(skip_bom: true)`
- naruse: If a code rewind and write BOM, this change will break such code.
- Let `CSV#rewind` know if it need to handle bom: test if the input is unicode and skip the bom???
  - Add `Encoding#unicode?` and `IO#skip_bom`
- => `rewind(skip_bom: true)`

[Feature #15822] Add Hash#except (zverok)

- The method is really useful, even with ActiveSupport-less codebases I constantly tend to redefine it with core ext or refinements. Hash#slice was merged in 2.5; it was initially discussed in #8499 alongside the except, but from the discussion, it is not obvious why #except was "lost" :(

Discussion:

ActiveSupport: https://api.rubyonrails.org/classes/Hash.html#method-i-except

```
# File activesupport/lib/active_support/core_ext/hash/except.rb, line 12
```

```
def except(*keys)
  slice(*self.keys - keys)
end
```

- knu: Use cases: `super(**options.expect(:foo))`, `json.except("metadata")`, `params.except(:id)`, etc.
- naruse: `Hash#slice` was introduced in the context of #13563, not #8499. In #13563, `Hash#except` is not discussed. That's the reason why #except was "lost".
- matz: Want to see the actual use case. And I don't like the name "except".
- Q. What about extending select/reject to alternatively take keys instead of a block?
    - A. (by knu) `hash.select()`/`hash.reject()` currently returns an Enumerator, so `hash.select(*keys)` wouldn't work as intended (returning a slice of hash) when `keys` were empty and that'd be confusing and error prone.

[Feature #16131] Remove $SAFE, taint and trust (jeremyevans)

- Does anyone have time to review the patch (extensive changes, mostly code deletion)?
- How do we want to handle included libraries with separate upstreams that want to be compatible with older Ruby versions (bundler, rubygems, etc.)?

Discussion:

- akr: Adding new mechanism to make warnings for transitions would make sense (the warnings are not printed with `-v`.)
- No recommendation from ruby committers to rubygems team.
- mame: Will reply.

[Feature #16255] Make monitor.rb built-in (ko1)

- maybe there is no problem.
- Should we make MonitorMixin built-in?

Discussion:

- matz: to make built-in expresses the wrong message to users to encourage to use threads
- ko1: ok, make it as extention library
- ko1: is `Thread::Monitor` necessary?
- usa: if it's built-in, it's necessary. but not, no.

[Feature #16254] MRI internal: Define built-in classes in Ruby with intrinsic syntax (ko1)

- only on small start, maybe there is no problem.
- many points we need to decide.

Discussion:

TracePoint: https://gist.github.com/ko1/969e5690cda6180ed989eb79619ca612

Comparable: https://gist.github.com/ko1/7f18e66d1ae25bb30c7e823aa57f0d31

- Need two files?
- Boot time performance.
- miniruby dependencies to external files.
- `__intrinsic__` doesn't look great => keep using `__intrinsic__` for now.
- matz: no consideration because it's an implementation detail but bigger than my expectation.
- akr: How about conditional compilation? We can use #ifdef freely now but Ruby file cannot refer such C-level constants.
- naruse: should release with 2.7.
- How will the ruby file path look like from ruby? => same as prelude `internal:trace_point.rb`.
- Timeline: will be merged within a few weeks, before preview 3.

[Feature #16253] Shorthand "forward everything" syntax (mame)

- How about introducing `def foo(...); bar(...); end`? It does not solve the compatibility issue of keyword argument separation, but it would provide a useful shorthand for delegation if the code doesn't have to work on 2.6.

Discussion:

- You are allowed to use `...` as many times as you want in one method body.

```
def parse_args(*args)
  args
end
def foo(...)
  args = parse_args(...)
  bar(...)
End
```

- block parameter cannot be `...` (implementation limitation)

```
  synchronize do |...| # SyntaxError
  end
```

- OK:
  - Valid only if `...` appears as a method parameter and it is parenthesized.
  - Can be used multiple times: `def foo(...); bar(...); baz(...); end`
  - `p = proc {|a,b,c| }; p.call(...)`
  - `def foo(...) super(...) end`
  - `def foo(...); def bar(...); baz(...); end; end`
  - `foo.[](...)`
  - `p.(...) # p.call(...)`
  - `def foo(...) iter{bar(...)}; end`
  - `def foo(...) end` # no error even if not used
  - `def foo(...); proc { bar(...) }; end; foo.call()`
- NG: (at least in the initial implementation)
  - Unavailable for block parameters.
  - `def foo(x, y, z) bar(...); end` # Usable only if declared in the parameter list.
  - `def foo(x, y, z) super(...); end` # ditto
  - `yield(...)` (because yield does not accept a block)
  - `return (...)` (ditto)
  - Use as value

        - `a = ...`
        - `[...]`
    - `p = proc {|a,b,c| }; p[...]`
    - `Array[...]`
    - `foo[...] = x`
    - `foo(...) { }` (can't call with a block)
    - `define_method(:foo) { |...| super(...) }`
    - `iter{|...|}`
    - `def foo(...): defined?(...); end` # syntax error
    - Without `()`: `foo ...` (becomes Range)
- OK if implementable
    - `def foo(...); eval('bar(...)'); end`
- Misc:
    - No future for begin-less && end-less range.
        - `...` should be meaningless; one could use `..` instead.
    - arity is `-1` just like `def f(*)`
    - parameters:
        - FYI: `p proc{}.parameters #=> []`
        - FYI: `def f(*) end; method(:f).parameters #=> [[:rest]]`
        - `def f(...) end; method(:f).parameters #=> ?`
    - `def foo(...) instance_eval(&Proc.new); end; def bar(...) foo { p(...) }; end; bar`
- Future work: lead argument handling is postponed
    - lead arguments can be extracted
    - lead arguments can be added
        - `def f(x, y, ...); g(1, 2, ...); end`

[Feature #16150] Add a way to request a frozen string from to_s (mame)

- `Symbol#to_s` started to return a frozen string in 2.7.0-preview1, and that revealed at least seven incompatibility issues. To make sure: is it okay to keep this change in 2.7.0-preview2?
- Issues reported from some gems including middleman, tested with trunk.
- More issues will be reported after preview2.

Discussion:

- mame: Fixing the issues is not very difficult.

- amatsuda: Rails fixed.
- naruse: negative. If unmaintained gems had a problem with this, what should one do?
- matz: Go ahead anyway. Get them fixed.

[Feature #16120] Omitted block argument if block starts with dot-method call (Dan0042)

- last time Matz said “Give me time to consider it”; there is now a patch ready; is it ok to accept?

Discussion:

- matz: No, I decided to reject it. Use the *_n* parameter.

[Feature #13083] {String|Symbol}#match{?} with nil returns falsy as Regexp#match{?} (znz)

- Matz said “Those methods (but =~) should consistently raise exceptions.” and pull request’s conflicts resolved again.

Discussion:

- matz: go ahead

[Misc #16258] Combine call info and cache to speed up method invocation (alanwu)

- Looking for feedback and reviews. I think this offers a good perf boost.

Discussion:

- ko1: will consider and reply

— time up —

[Bug #15912] Allow some reentrancy during TracePoint events (alanwu)

- I have seen quite a few people confused about Byebug's REPL not working as they would expect due to this issue. I think it's important to come up with some solution for this in two seven.

Discussion:

[Feature #16142] Implement code_range in Proc and Method - Ruby master - Ruby Issue Tracking System (osyo)

- Propose an API to get code position of Proc and Method so that we can get body of them (especially of a Proc).
- I want to discuss method names and return values

Discussion: