RBS Tutorial

RubyWorld Conference 2023
9th, Nov. 2023

pp self

- Masataka Pocke Kuwabara
- Work for Money Forward, Inc.
    - クラウド会計Plus
    - A maintainer of RBS
- Live in Okayama

Matsue and me

Goal

- You can start developing applications with RBS
    - without confusion.
- You can find references to learn RBS.

Agenda

- Describe `.rbs` files
    - The difference from .rb files
- RBS Syntax Overview
- How to start RBS
- Libraries
- References

This talk is based on the premise that it uses RBS and Steep.

What is .rbs file

What is .rbs file

It defines static types for `` `.rb` `` file.

- It is separated from `` `.rb` `` file.
- `` `.rbs` `` is not a Ruby file. RBS has different syntax from Ruby.

Separated from .rb file

- Basically, `` `.rb` `` file does not contain type information.
  - All type definitions have to be written in RBS files.
- It is similar to (`.c` and `.h`) or (`.js` and `.d.ts`).
- RBS environment is created only from `.rbs` file.
  - If `.rb` file contains a class definition, RBS does not recognize the class without definition in `.rbs` file.

In the future (nothing determined)

- RBS may support writing RBS in `.rb` files
  - For example, as a comment (It is just an example!)

    ```
    class C
      # @rbs: (Integer) -> String
      def f(int) = int.to_s
    end
    ```
- RBS may relax unknown classes/modules/methods
  - For example, RBS can generate RBS definition from `.rb` files on runtime

RBS Syntax Overview

RBS Syntax

I'll describe RBS syntax overview.

It is similar to Ruby syntax, but it is different.

Classes / Modules

```ruby
# Ruby

module M
end

class C
  X = 42
  include M
end

C2 = C
```

```rbs
# RBS

module M
end

class C
  X: Integer
  include M
end

class C2 = C
```

Method Definitions

```
# Ruby

class C
  def f1 = 42
  private def f2(int) = int.to_s

  private

  def f3(&block) = (block.call; 42)
  def f4(x:) = x + 42

  attr_reader :x
end
```

```
# RBS

class C
  def f1: () -> Integer
  private def f2: (Integer int) ->
String

  private

  def f3: () { () -> void } ->
Integer
  def f4: (x: Integer) -> Integer

  attr_reader x: Integer
end
```

Interfaces

```ruby
# Ruby

class IO
  def read(...)= ...
end

class StringIO
  def read(...)= ...
end

def read_from_io(io) = io.read
```

```rbs
# RBS

interface _Reader
  def read: (
    ?int? length,
    ?string outbuf
  ) -> String?
end

def read_from_io: (_Reader io) ->
String?
```

Other syntaxes

- Type Alias
- Type Parameter
- variables
    - (instance | class | class instance | global) variables
- `use` directive
- ...and more!

See the following documentation for more information.
https://github.com/ruby/rbs/blob/master/docs/syntax.md

How to start RBS

How to start RBS

I'll describe a minimal example to start developing an app with RBS / Steep and VS Code

Editor supports

- Many editors support RBS / Steep
- Technically, Steep works on editors supporting LSP

See the full list of editors supporting RBS
https://github.com/ruby/rbs/blob/master/docs/tools.md

VS Code for RBS

I recommend using VS Code because

- VS Code is well-integrated to LSP
- It has officially developed plugins to integrate RBS
  - https://marketplace.visualstudio.com/items?itemName=soutaro.rbs-syntax
  - https://marketplace.visualstudio.com/items?itemName=soutaro.steep-vscode

Other editors also support RBS, you can use your favorite editor💕

Prepare gems

Add Steep gem to your Gemfile

```
gem "steep", require: false
```

And run `bundle install`

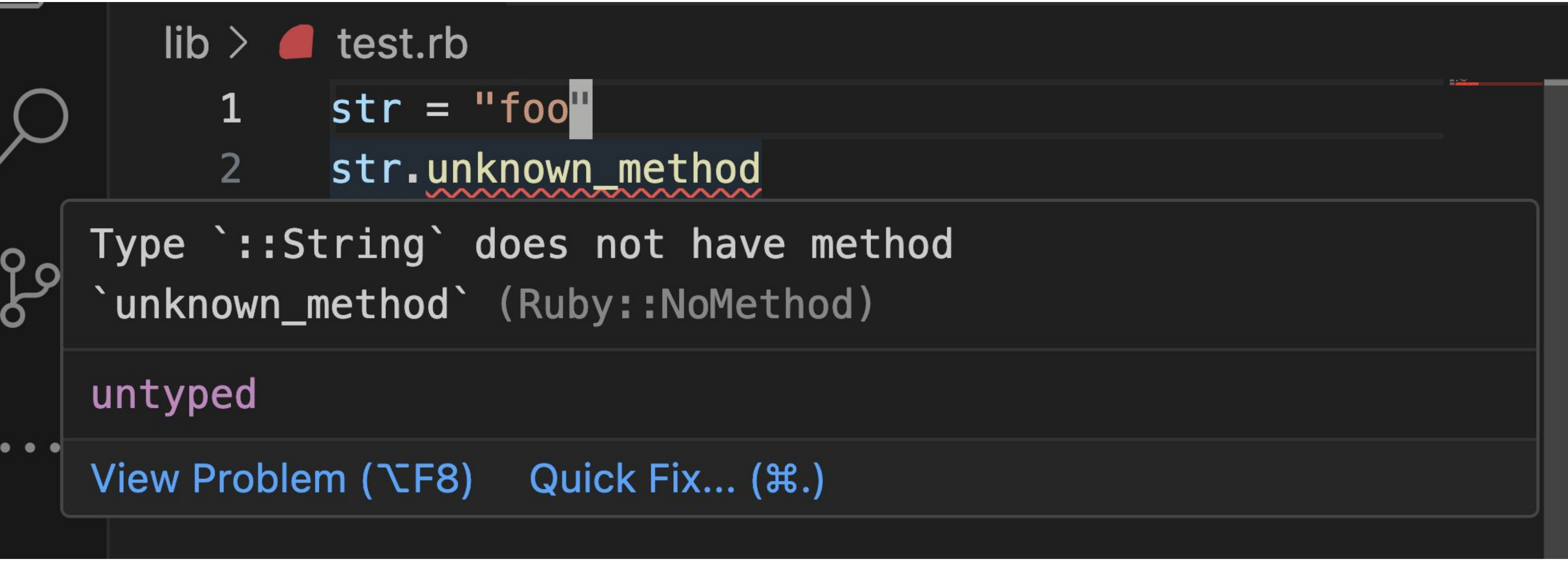
Note that `gem "rbs"` is not required because Steep depends on RBS gem.

Minimum configuration of Steep

```
# Steepfile

target :lib do
  signature "sig"  # Specify where .rbs files are in
  check "lib"      # Specify where .rb files are in
  check "app"      # For Rails app
end
```

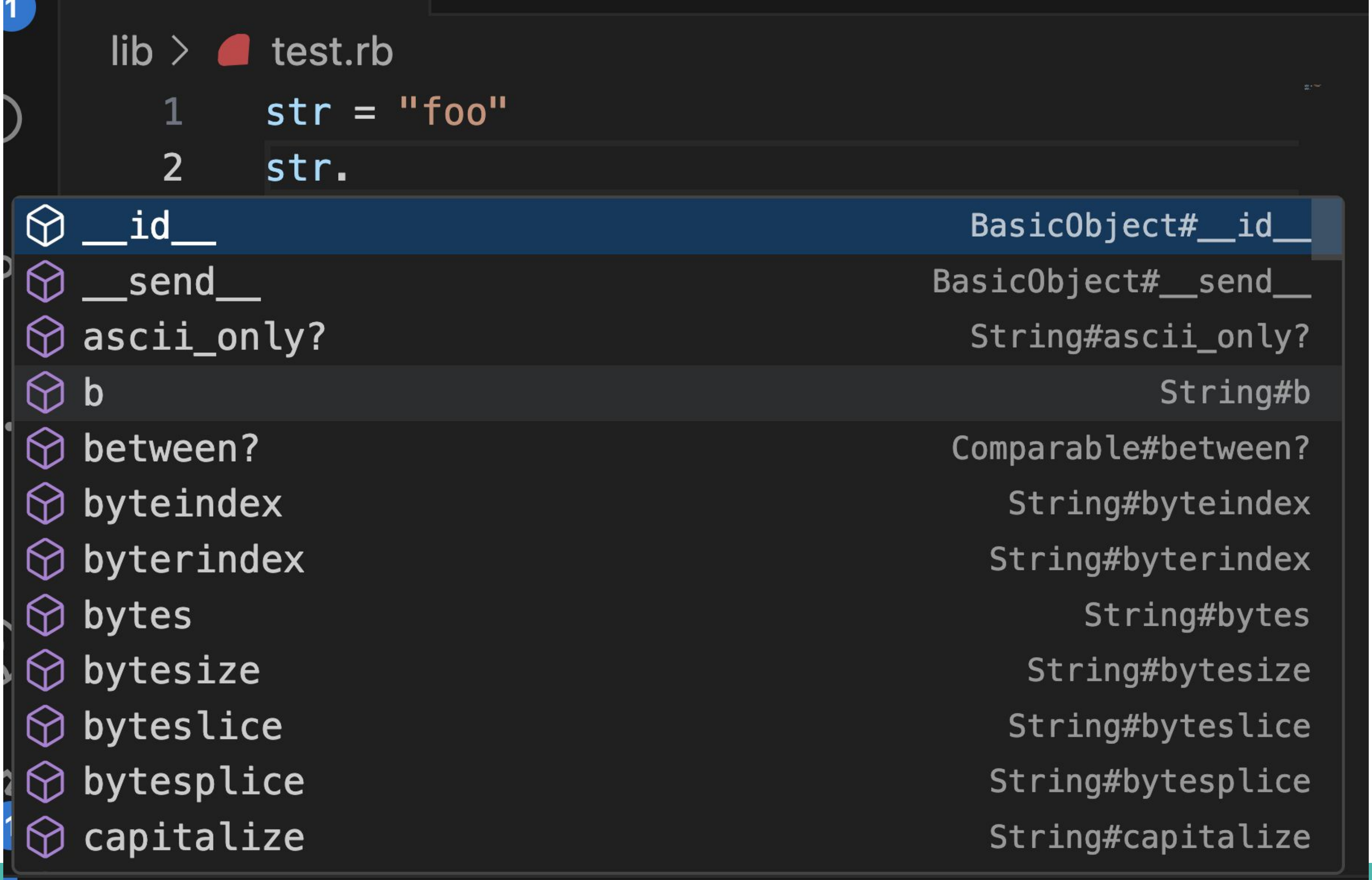
For more information, see `Steepfile` generated by `steep init`.



```ruby
str = "foo"
str.unknown_method
```

```
Type `::String` does not have method
`unknown_method` (Ruby::NoMethod)
```

```
untyped
```


lib > test.rb

```ruby
str = "foo"
str.
```

```
__id__          BasicObject#__id__
__send__        BasicObject#__send__
ascii_only?     String#ascii_only?
b               String#b
between?        Comparable#between?
byteindex       String#byteindex
byterindex      String#byterindex
bytes           String#bytes
bytesize        String#bytesize
byteslice       String#byteslice
bytesplice      String#bytesplice
capitalize      String#capitalize
```

Directory structure

- You should put RBS files under `` `sig/` `` directory
  - In gem package development, RBS files under the directory are exposed.
- No restriction of directory structure under `` `sig/` ``
  - But I recommend using the same directory structure as the `` `.rb` `` files.
  - In a Rails app: `app/models/user.rb : sig/models/user.rbs`
  - In a gem: `lib/foo/bar.rb : sig/foo/bar.rbs`

Tips: bin/steep

- Steep VS Code plugin supports `bin/steep` executable file.
- If `bin/steep` is available, the plugin uses it instead of `bundle exec steep`
- If you need to configure `steep` command, you can use this file.

For larger applications

- You can use RBS Rails gem for a Rails application
    - https://github.com/pocke/rbs_rails
- You can use RBS generator, such as `rbs prototype`, to generate RBS of existing Ruby code.
- For more information, check out my talk at RubyKaigi 2023
    - https://rubykaigi.org/2023/presentations/p_ck_.html#day3
    - This talk has demonstration, describing tools such as `rbs subtract`, for large app.

Libraries

Kinds of Libraries

- Core Library
  - It is installed by default and loaded by default (no require necessary)
  - Example: Array, String, etc...
- Standard Library
  - It is installed by default, but you need require to load it.
  - It includes default gems.
  - Example: pathname, ripper, etc...
- Gem
  - Other gems, including bundled gems.
  - Example: activerecord, nokogiri, etc...

See https://stdgems.org/ for the definition of (default / bundled) gems.

Core Library (Array, String, etc...)

RBS provides core libraries types out of the box

- In Ruby, we do not need require to use a core library
- Then, in RBS, we do not need to do anything to use a core library

Standard Library (pathname, ripper, etc...)

RBS gem contains their signatures

- You do not install anything except rbs gem
- But you need to specify the gem explicitly to load it
  - You can use `rbs collection` for this purpose

Gem

RBS can load third party gems RBS files from

- `sig/` directory in gem package
- GitHub repository, ruby/gem_rbs_collection

Library management: rbs collection

`rbs collection` manages RBSs of gems

- `rbs collection install` installs RBS files depended by your application
  - It resolves the dependency from `Gemfile.lock`.

Check my previous talk in RubyKaigi 2021 Takeout for more details

https://rubykaigi.org/2021-takeout/presentations/p_ck_.html

References

Syntax

- Official document: https://github.com/ruby/rbs/blob/master/docs/syntax.md
- My Blog articles:
  - https://pocke.hatenablog.com/entry/2021/01/02/175940 (2y ago)
  - https://moneyforward-dev.jp/entry/2023/10/13/rbs-new-syntaxes (only new syntaxes)
- For developers:
  - https://github.com/ruby/rbs/blob/master/ext/rbs_extension/parser.c

Official documents

See `docs/` directory https://github.com/ruby/rbs/tree/master/docs

- I recommend the following document for beginners
  - https://github.com/ruby/rbs/blob/master/docs/rbs_by_example.md
- You can find editor integrations from this document
  - https://github.com/ruby/rbs/blob/master/docs/tools.md

Existing RBSs

You can find `.rbs` files from the following places

- Core Libraries: https://github.com/ruby/rbs/tree/master/core
- Standard Libraries: https://github.com/ruby/rbs/tree/master/stdlib
- Gems: https://github.com/ruby/gem_rbs_collection

My Talks

- The newsletter of RBS (RubyKaigi Takeout 2021)
  https://speakerdeck.com/pocke/the-newsletter-of-rbs-updates
  - It mainly describes `rbs collection`.
- Let's write RBS! (RubyKaigi 2023)
  https://speakerdeck.com/pocke/lets-write-rbs
  - It mainly describes `rbs subtract`.

Thanks for your listening!