ZARD

Yet another YARDoc.



pixiv Inc.
USAMI Kenta

pixiv

2026-10-08 Proposals on Rails 2026
イタンジ株式会社 #roppongirb

お前誰よ



- うさみけんた（@tadsan）/ **Zonu.EXE** / **にゃんだーすわん**
- **ピクシブ株式会社** Platform Div > WebTechnology Team **PHPer**
  - 2012年末から現職、APIとかCIとかいろいろなところを見つめてきました
  - 最近チームが再編されてインフラっぽい仕事もしてます
- Emacs PHP Modeを開発しています（2017年-）
- プログラミング言語にちょっとこだわりのある素人（spcamp2010）

はじめましての方は
はじめまして

PIXIV FANBOX

クリエイター



にゃんだーすわん

クリエイターページ

投稿する

ダッシュボード

投稿管理

プラン管理

クリエイター設定

ファン一覧

おたより

支援金管理/振込

FANBOXプリント

FANBOXプリントとは

コンテンツ提供者申請

クリエイター優待



ちょっとすごいRubyの型チェッカーを作ってます

編集

2026年5月25日 23:20・全体公開

これは自画自賛しても許されるくらいちゃんとすごい技術だと思っているのですが…

私はLispやRubyのような動的言語が大好きで、2016年くらいからPHPに型をつけながら暮らしている者です。もともとは2010〜2014年くらい前までShibuya.rbやRuby会議に顔を出していたものの、仕事でPHPをやっていることもあってPHP方面に軸足を移していました。

pixiv

Rigor 検索 ⌘K

Rigor

Rubyソースに注釈を書かなくても、フローに応じた実用的な型フィードバックを返す静的解析器です。

インストールして試す → ハンドブックを読む

ブラウザで試す ソースを見る

English
Read in English

日本語
日本語で読む

最新リリース v0.4.0 · 2026年9月27日

Rigorはソースに型注釈を一切書かずに`rigor check`でRubyコードを解析し、型付きレシーバーへの未定義メソッド呼び出し、引数個数の誤り、証明可能なnilデリファレンスなどの実際のバグを報告します。型はすべてコードが実際に生成する値から推論されます。チームに型注釈を書いてもらう必要はありません。

pixiv

なぜ新しい型チェッカーを書きはじめたのか

How I Ended Up Writing a New Type Checker for Ruby



pixiv Inc.
USAMI Kenta

pixiv

2026-09-05 #rubykaigi followup
RubyKaigi 2026 follow up

pixiv

PHPの「型」は どこまで戦えるか

Can PHP Types Still Compete?
Pushing the Limits of Static Analysis.



pixiv Inc.
USAMI Kenta

pixiv

2026-10-03 #phpcon ehime
PHPカンファレンス愛媛 @愛媛大学 城北キャンパス

pixiv

便利なので
使ってください

今回はその話ではない

pixiv

現代の生成AIは
大規模言語モデル

コーディングエージェントがいかに賢く振る舞っても確率からは逃れられない

モデルが持っている常識
×
プロジェクトの雰囲気

水は低きに流れる

AIは常識を振りかざす前例主義者

何が言いたいか

大規模言語モデル

生成AIは言語で
あれこれ語りたがる

コメントが多い
プロジェクトは
自然に雄弁になる

プログラミング言語
自然言語

コメントは
コードで表現できない
行間を補う機能

パラメータ名で表現しきれない構造の評価

歯止めを掛けないと
AIはコードに
ポエムを書き散らす
ポエマーになる

コードにはあくまで
行間を書いてほしい


Post
Takuto Wada
@t_wada
Show translation
コードには How
テストコードには What
コミットログには Why
コードコメントには Why not
を書こうという話をした
12:56 PM · Sep 5, 2017
13
3.6K
11K
6.1K

pixiv

「自分の未来予測を信じてちょっと意地を張ってみる」　まつもとゆきひろ氏がRubyに型宣言を入れない理由

コピー　ブックマーク



pixiv

型宣言がなくても静的型宣言のメリットは実現できる

例えば、Ruby3.0から静的型チェックを導入するようになったんですけれども、型宣言はなかったんですね。つまり、型情報はそのプログラムとは別のところに持っていて、その情報を使って型チェックを行おうと。ないものについては推論をしようというアプローチですね。

もちろん、今のRubyの静的型チェックは誕生したばかりでまだ未成熟なんですけれども、原理的には、型宣言なしで型チェックの恩恵をある程度受けることができると言われています。がんばっている途中ですね。ただ、型宣言はないので、ドキュメントとしての役割は果たせません。

ただ、そうはいっても型情報はいろいろあるので、そこによる推論結果みたいなものをIDEで表示することによって、ドキュメントとしての役割をある程度果たせますし、Rubyには、インラインドキュメント。つまり型がなくてもいずれにしても振る舞いは書かないといけないのでインラインドキュメントをする習慣があって、型情報を書いたらそれを取り込んでくる。型情報として取り込んでくるというツールも今開発中です。

Rubyに型宣言を入れない理由

ただ、一度言語に型宣言を持たせると、もう取り除けないわけですよ。型宣言のあるプログラムがもうそこにたくさんできちゃうわけです。さらに、コミュニティの中におそらくですが、型警察みたいなものが出てくると思うんですね。

ジェムとかライブラリを作っている人のところに行って、「お前の使っているこのライブラリに型宣言がないんだけど、型宣言は絶対にあるべきだ」みたいに言って、型宣言を付けていないプログラムに対して文句をつけまくるみたいな人たちがコミュニティに登場しそうな気がするんですね。実際によその言語ではそういうことが発生していると聞くんですね。

Zenn

投稿する



Claude Code に向いているプログラミング言語

2026/03/05に公開　12件

AI　プログラミング言語　ベンチマーク　llm　Claude Code　Tech

English version →

TL;DR

Claude Code に 13 言語で簡易 git を実装させたところ、Ruby・Python・JavaScript が最速・最安・最安定でした。静的型付けの言語は 1.4〜2.6 倍遅く、生成コストがかかる傾向がありました。

Cost for Claude Code to Generate a Mini-Git (v1+v2, 20 trials)



Yusuke Endoh

フォロー

World No.1 IOCCC player.
http://ioccc.org/winners.html#Yusuke_Endoh

バッジを贈る

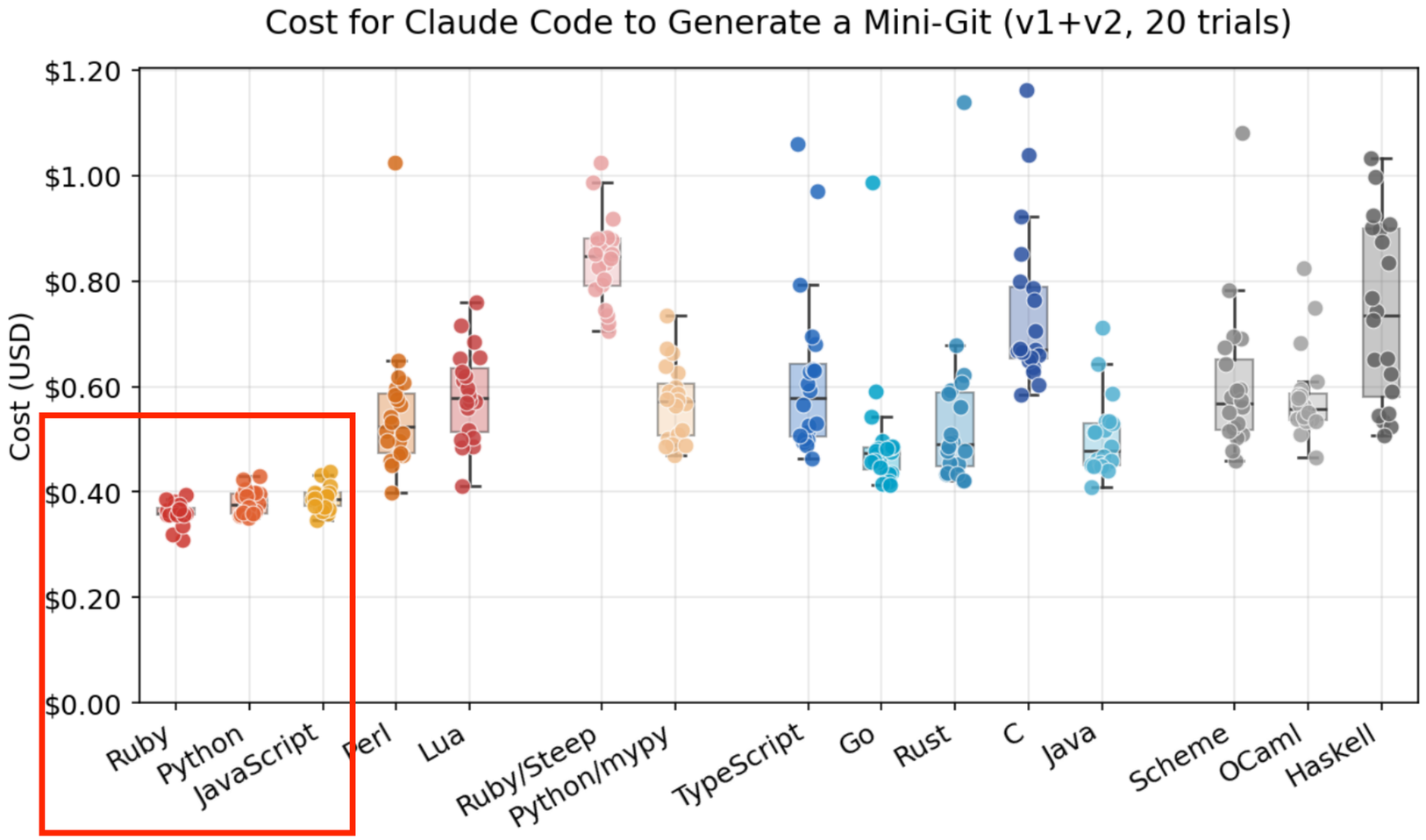
バッジを贈るとは →

目次

pixiv

TL;DR

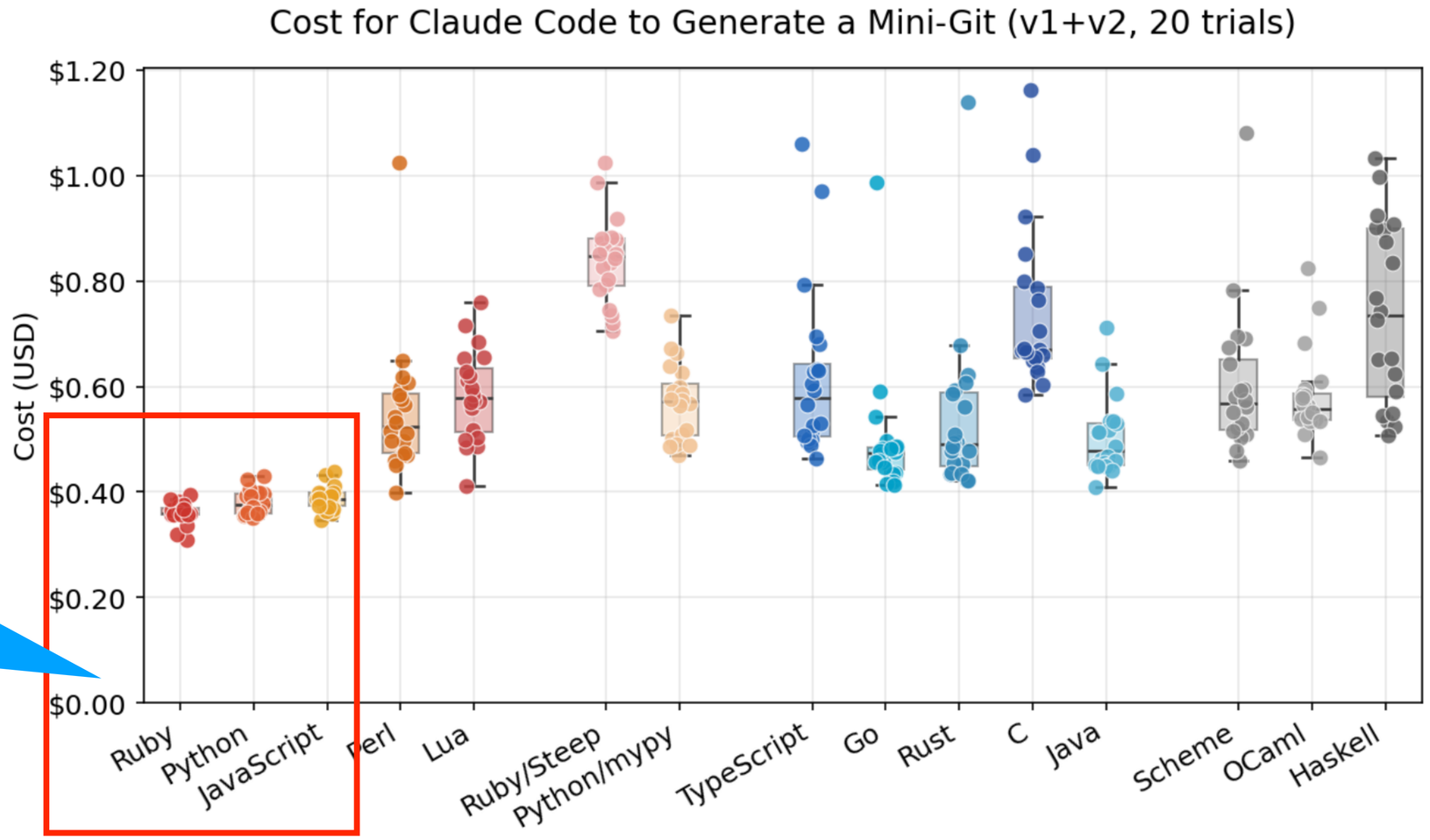
Claude Code に 13 言語で簡易 git を実装させたところ、Ruby・Python・JavaScript が最速・最安・最安定でした。静的型付けの言語は 1.4〜2.6 倍遅く、生成コストがかかる傾向がありました。



pixiv

TL;DR

Claude Code に 13 言語で簡易 git を実装させたところ、Ruby・Python・JavaScript が最速・最安・最安定でした。静的型付けの言語は 1.4〜2.6 倍遅く、生成コストがかかる傾向がありました。



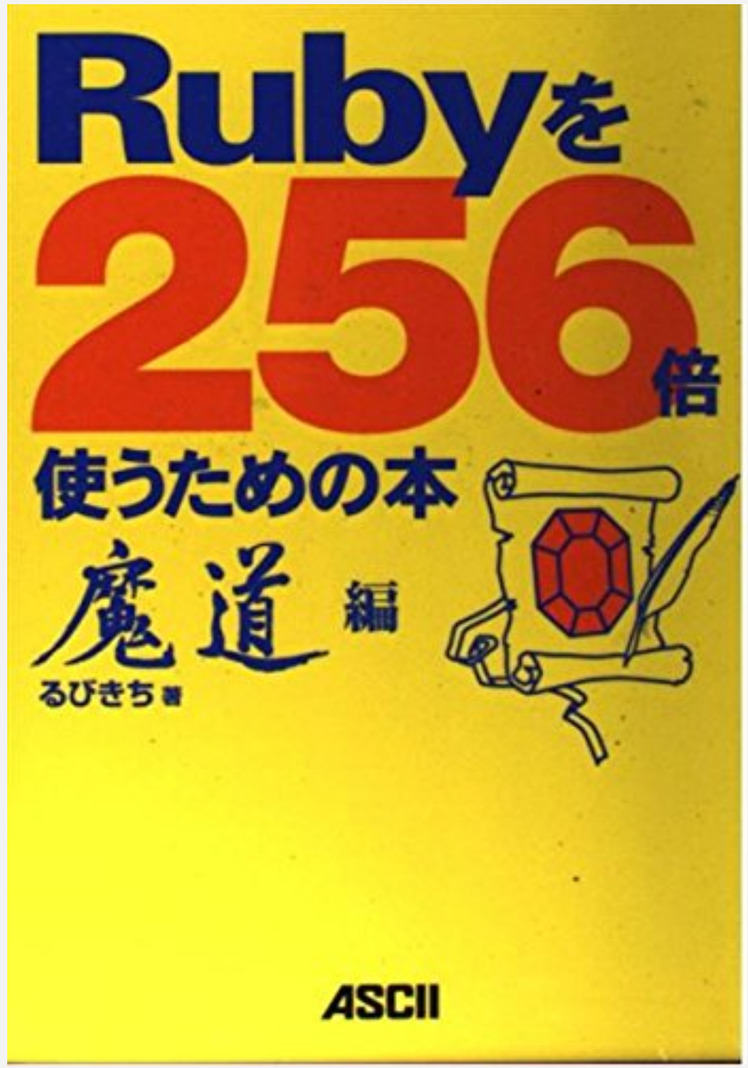
シンプル
型なし

pixiv

完全な散文よりも
構造化形式で
書いておきたい

Rubyでは
歴史的にいろんな
フォーマットがあった

RD



pixiv

RDoc

Ruby 4.0 リファレンスマニュアル latest クラス・メソッドを検索 (/)

Ruby 4.0 リファレンスマニュアル > ライブラリ一覧 > rdoc ライブラリ

library rdoc

[edit]

要約

RDoc は Ruby のドキュメント生成を行うためのライブラリです。rdoc というドキュメント生成のためのコマンドも含んでいます。

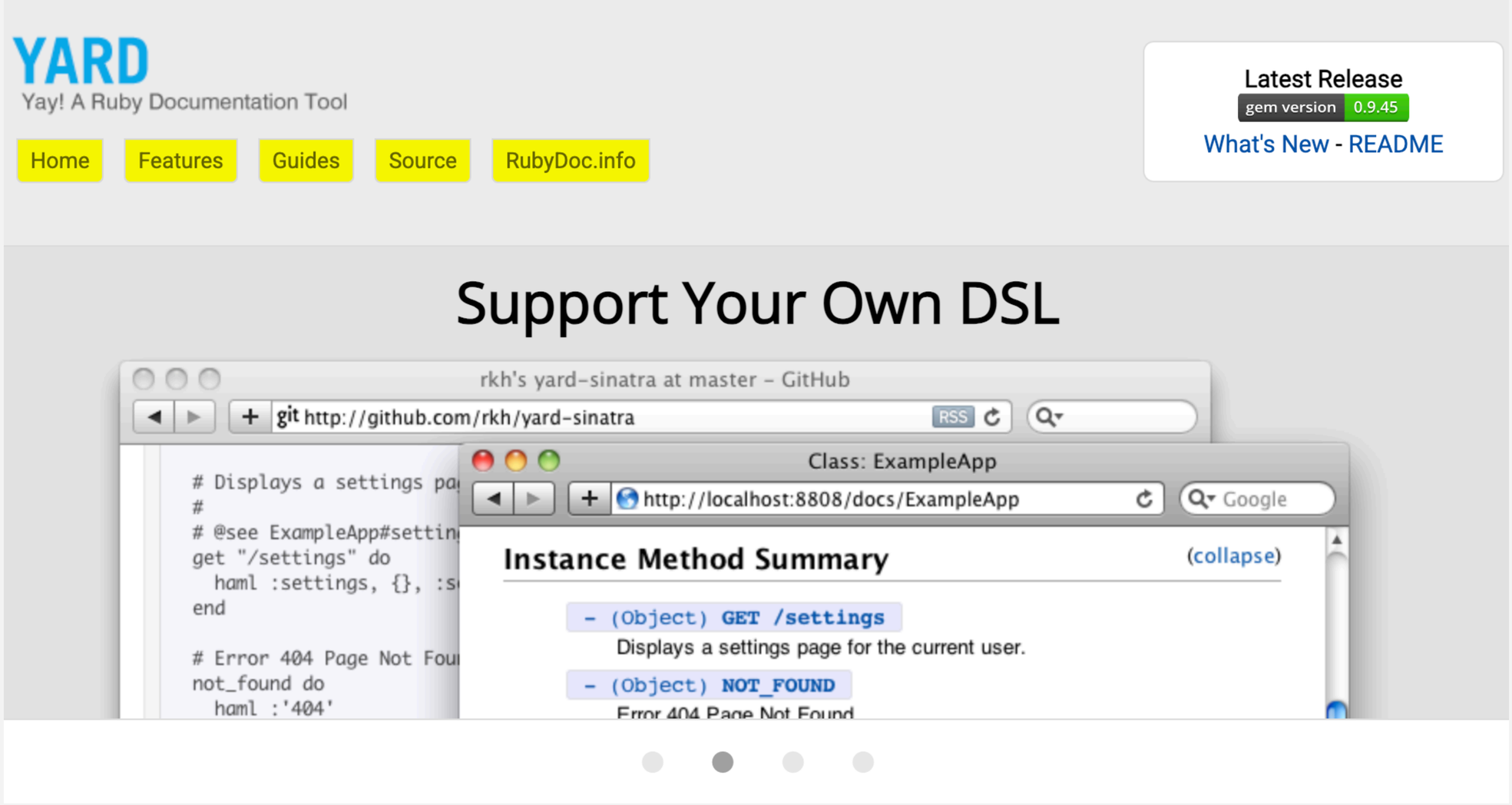
このパッケージは RDoc と Markup というふたつのコンポーネントを含んでいます。 RDoc とは Ruby のソースファイルに対するドキュメントを生成するアプリケーションです。 JavaDoc と同様に、ソースを解析し、クラス、モジュール、メソッドの定義を抜き出してきます(include,require もです)。そしてこれらの内容とその直前に書かれたコメントを併合し、ドキュメントを出力します(現在は HTML しか出力できませんが、この部分は取り替え可能にできています)。Markup とはプレーンテキストを様々なフォーマットに変換するためのライブラリです。RDoc によってメソッドやクラスに関するドキュメントを生成するとき、コメント部を変換するために使われます。

ロードマップ

- RDoc で Ruby のソースファイルに対するドキュメントを生成したければ、まずこの文章を読みましょう。
- C で書かれた拡張ライブラリを含めたければ、**rdoc/parser/c** を参照してください。
- コメント部で使えるマークアップについて知りたければ、**rdoc/markup** を参照してください。
- RDoc をライブラリとして使いたければ、**RDoc::RDoc** を参照してください。
- テキスト部を HTML に変換する部分をライブラリとして使いたければ、**RDoc::Markup** を参照してください。

pixiv

Yay! A Ruby Documentation Tool



pixiv

rbs-inline (method)

Method definitions

We have two syntaxes for type of methods.

`#:` syntax

This is more primitive syntax. It allows writing RBS method type embedded in Ruby code.

```
#: () -> String
def to_s
end

#: (?Regexp?) -> Enumerator[String, void]
#: (?Regexp?) { (String) -> void } -> void
def each_person(pattern = nil, &block)
end
```

Having multiple `#:` syntax generates method definition with overloads.

`# @rbs` syntax

You can also write method types after `# @rbs` keyword.

```
# @rbs () -> String
def to_s
end

# @rbs (?Regexp?) -> Enumerator[String, void]
#    | (?Regexp?) { (String) -> void } -> void
def each_person(pattern = nil, &block)
end
```


pixiv

rbs-inline (Doc style)

Doc style syntax

Here is the example of doc style syntax.

```
# @rbs size: Integer -- The size of the section in px
# @rbs optional: Integer -- Type of the optional parameter
# @rbs title: String -- Title of the section
# @rbs content: String -- Type of the optional keyword parameter
# @rbs *rest: String -- Type of the rest args
# @rbs **kwrest: untyped -- Type of the keyword rest args
# @rbs return: Section? -- Returns the new section or `nil`
def section(size, optional=123, title:, content: "Hello!", *rest, **kwrest)
end
```

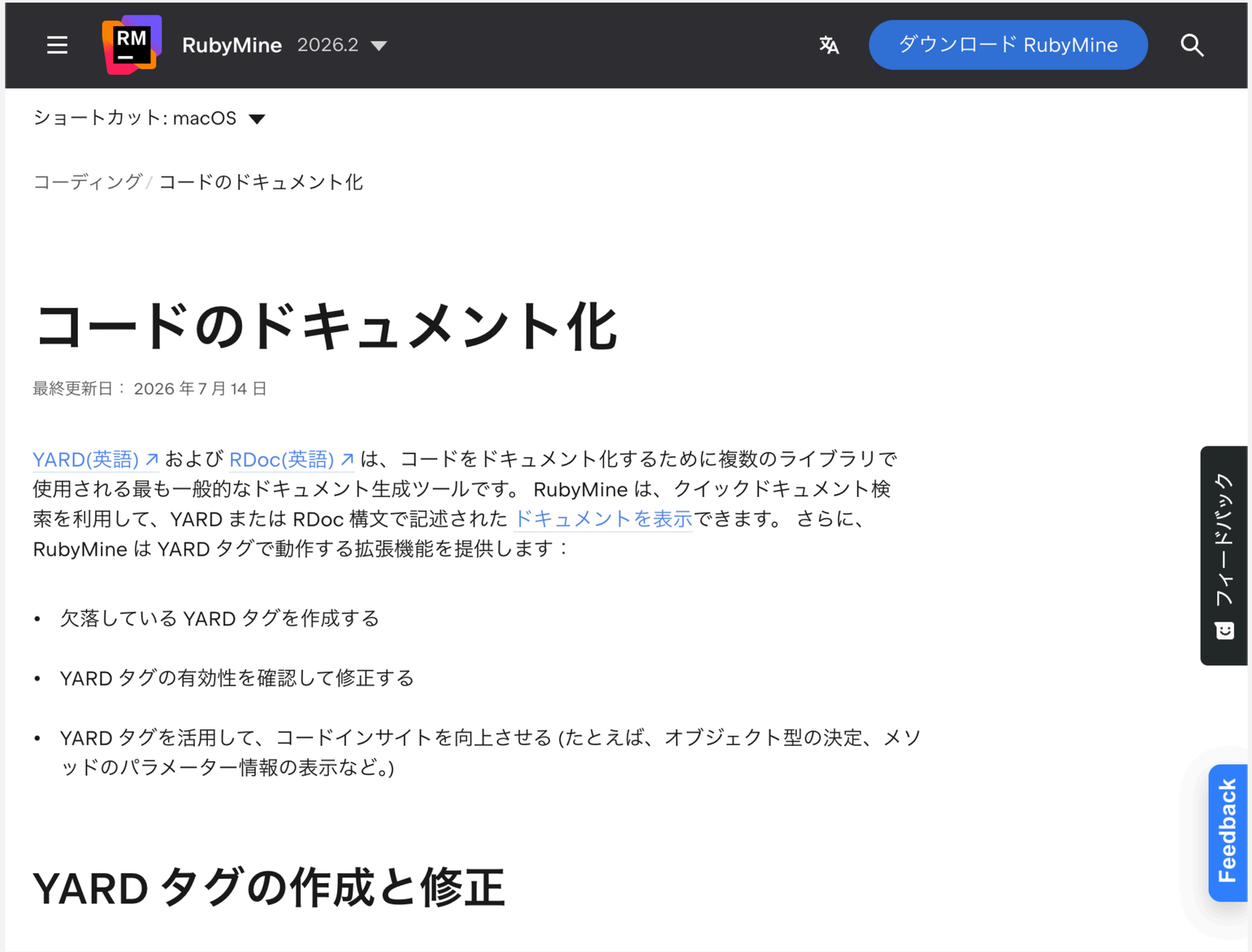
You can write down type of parameters like `# @rbs var: TYPE -- some comments`. The `@rbs return: TYPE` defines the return type.

The block types can be defined with `# @rbs &block:` annotations.

```
# @rbs &block: (?) -> untyped -- Block is required, but not clear what will be yielded
def foo(&block)
  yield 1
  yield 2, "true"
end
```


pixiv

RubyMine



pixiv

Rigorの思想

人間に冗長な型を
わざわざ
明記させない

pixiv

型推論して
RBSを生成して
ガードレールにする

必要以上に詳細に型を書かせない

AIでちょんと
セットアップすれば
静的解析できる！

Rigorによる持続可能なRailsの静的型解析

Speaker

USAMI Kenta

Format

Short session (15 min)

Track

General

Abstract

筆者が開発する「Rigor」は、人力での運用と型の知識をほとんど必要としない、Rubyのための新しい静的解析ツールです。

Rubyにおける型エコシステムが発展を続ける一方、導入を検討したものの「型定義管理の重さ」「ActiveRecordをはじめとするDSLやメタプログラミングとの相性」「チームメンバーへの学習コスト」といった現実的な運用の壁に直面することも少なくないかと思います。

筆者が提案するのは、手動による型定義のメンテナンスを避け、型推論による決定論的な型定義の生成とコーディングエージェントを活用することで、プロジェクトの実情に応じた的確なセットアップを半自動で行うアプローチです。

Rigorは既に「動作している」実績を持つRubyコードに対して、過剰な警告を避けながら徹底的な型推論を行います。型定義を手動でメンテナンスすることなく、AI時代においてもRubyの良さをスポイルせずに「ガードレール」としての型を享受できるよう設計されています。

本セッションでは、Rigorの誕生背景と現実のRailsプロジェクトへ型理論を適用する具体的なアプローチについてご紹介します。

pixiv

Rigorによる持続可能なRailsの静的型解析

Speaker

USAMI Kenta

Format

Short session (15 min)

Track

General

Withdraw Proposal

Status: NOT ACCEPTED Updated: 3ヶ月 ago

Abstract

筆者が開発する「Rigor」は、人力での運用と型の知識をほとんど必要としない、Rubyのための新しい静的解析ツールです。

Rubyにおける型エコシステムが発展を続ける一方、導入を検討したものの「型定義管理の重さ」「ActiveRecordをはじめとするDSLやメタプログラミングとの相性」「チームメンバーへの学習コスト」といった現実的な運用の壁に直面することも少なくないかと思います。

筆者が提案するのは、手動による型定義のメンテナンスを避け、型推論による決定論的な型定義の生成とコーディングエージェントを活用することで、プロジェクトの実情に応じた的確なセットアップを半自動で行うアプローチです。

Rigorは既に「動作している」実績を持つRubyコードに対して、過剰な警告を避けながら徹底的な型推論を行います。型定義を手動でメンテナンスすることなく、AI時代においてもRubyの良さをスポイルせずに「ガードレール」としての型を享受できるよう設計されています。

本セッションでは、Rigorの誕生背景と現実のRailsプロジェクトへ型理論を適用する具体的なアプローチについてご紹介します。

pixiv

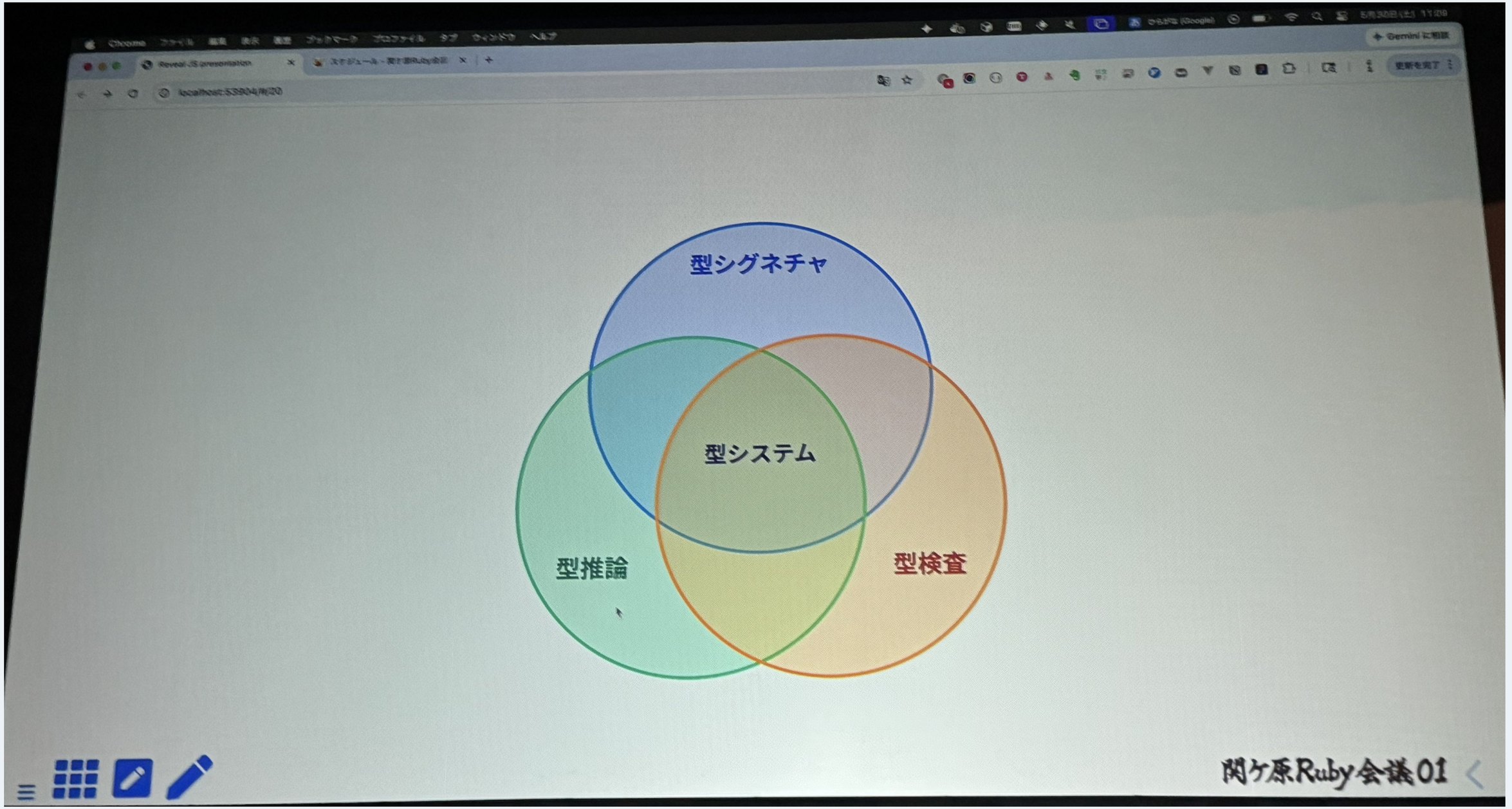


関ケ原Ruby会議01

拙者、
『型は欲しいが型は書きたくない』
者たちとの和睦を結び、
るびぃにおける型の領地安堵を
実現せんと欲す者也

大江森塚三矢大坂守小輔次郎真年
@sanfrecce_osaka
2026/05/30 #sekigahara01

pixiv


型シグネチャ
型システム
型推論
型検査
関ケ原Ruby会議01

型シグネチャはドキュメントの表現として非常に有用

yard のタグよりも rbs の方が表現力が高い

```
# こっちより
#
# @param bar [Hash{Symbol=>String,Integer}] key は uuid・name・age で age は optional、uuid は UUIDv4 形式
# @return [Foo] ○○ な ｘｘ
def foo(bar)

# こっちの方が表現力が高い
#
# @rbs bar: { uuid: String, name: String, ?age: Integer } -- uuid は UUIDv4 形式
# @rbs return: Foo -- ○○ な ｘｘ
def foo: ({ id: Integer, name: String, ?age: Integer } bar) -> Foo
```

関ケ原Ruby会議01

pixiv

Rubyに型を
書きたくないのに
Docに型を書かされる

ZARD

Yet another YARDoc.



pixiv Inc.
USAMI Kenta

pixiv

Syntax at a Glance

ZARD separates type contracts from API prose:

```ruby
# @extrbs return: non-empty-string
# @rbs path: String
# @rbs return: String
# @param path — Path to the name file.
# @return The stored name.
# @raise IOError — If the file cannot be read.
def read_name(path)
  File.read(path).strip
end
```

- **Contracts**: Plain RBS uses `@rbs` or `#:`. Advanced Rigor-only refinements use `@extrbs`. Contract payloads are not copied into API documentation. Inline RBS trailing prose after `--` is stored as a contract note, not API prose.
- **Documentation Marker**: An em dash (`—`) separates named documentation tags (`@param`, `@yieldparam`, `@raise`, `@option`) or optional type claims from descriptions.
- **Tags without names or type claims**: Tags such as `@return`, `@note`, and `@example` write their descriptions directly without an em dash.
- **Continuations**: Plain comment lines continue documentation text until the next annotation or contract.
- **Encoding**: ZARD documentation requires UTF-8 source text.

pixiv

Syntax at a Glance

ZARD separates type contracts from API prose:

型は @rbs
または #: に書く

```ruby
# @extrbs return: non-empty-string
# @rbs path: String
# @rbs return: String
# @param path — Path to the name file.
# @return The stored name.
# @raise IOError — If the file cannot be read.
def read_name(path)
  File.read(path).strip
end
```

- **Contracts**: Plain RBS uses `@rbs` or `#:`. Advanced Rigor-only refinements use `@extrbs`. Contract payloads are not copied into API documentation. Inline RBS trailing prose after `--` is stored as a contract note, not API prose.
- **Documentation Marker**: An em dash (`—`) separates named documentation tags (`@param`, `@yieldparam`, `@raise`, `@option`) or optional type claims from descriptions.
- **Tags without names or type claims**: Tags such as `@return`, `@note`, and `@example` write their descriptions directly without an em dash.
- **Continuations**: Plain comment lines continue documentation text until the next annotation or contract.
- **Encoding**: ZARD documentation requires UTF-8 source text.

pixiv

Syntax at a Glance

ZARD separates type contracts from API prose:

```ruby
# @extrbs return: non-empty-string
# @rbs path: String
# @rbs return: String
# @param path — Path to the name file.
# @return The stored name.
# @raise IOError — If the file cannot be read.
def read_name(path)
  File.read(path).strip
end
```

型は @rbs
または #: に書く

@param @return
にはコメントだけを書く

- **Contracts**: Plain RBS uses `@rbs` or `#:`. Advanced Rigor-only refinements use `@extrbs`. Contract payloads are not copied into API documentation. Inline RBS trailing prose after `--` is stored as a contract note, not API prose.
- **Documentation Marker**: An em dash (`—`) separates named documentation tags (`@param`, `@yieldparam`, `@raise`, `@option`) or optional type claims from descriptions.
- **Tags without names or type claims**: Tags such as `@return`, `@note`, and `@example` write their descriptions directly without an em dash.
- **Continuations**: Plain comment lines continue documentation text until the next annotation or contract.
- **Encoding**: ZARD documentation requires UTF-8 source text.

pixiv

Syntax at a Glance

ZARD separates type contracts from API prose:

```ruby
# @extrbs return: non-empty-string
# @rbs path: String
# @rbs return: String
# @param path — Path to the name file.
# @return The stored name.
# @raise IOError — If the file cannot be read
def read_name(path)
  File.read(path).strip
end
```

型は @rbs
または #: に書く

@param @return
にはコメントだけを書く

@extrbs という
怪しい型も書ける

- **Contracts**: Plain RBS uses `@rbs` or `#:`. Advanced Rigor-only [illegible] `@extrbs`. Contract payloads are not copied into API documentation. Inline RBS trailing prose after `--` is stored as a contract note, not API prose.
- **Documentation Marker**: An em dash (`—`) separates named documentation tags (`@param`, `@yieldparam`, `@raise`, `@option`) or optional type claims from descriptions.
- **Tags without names or type claims**: Tags such as `@return`, `@note`, and `@example` write their descriptions directly without an em dash.
- **Continuations**: Plain comment lines continue documentation text until the next annotation or contract.
- **Encoding**: ZARD documentation requires UTF-8 source text.

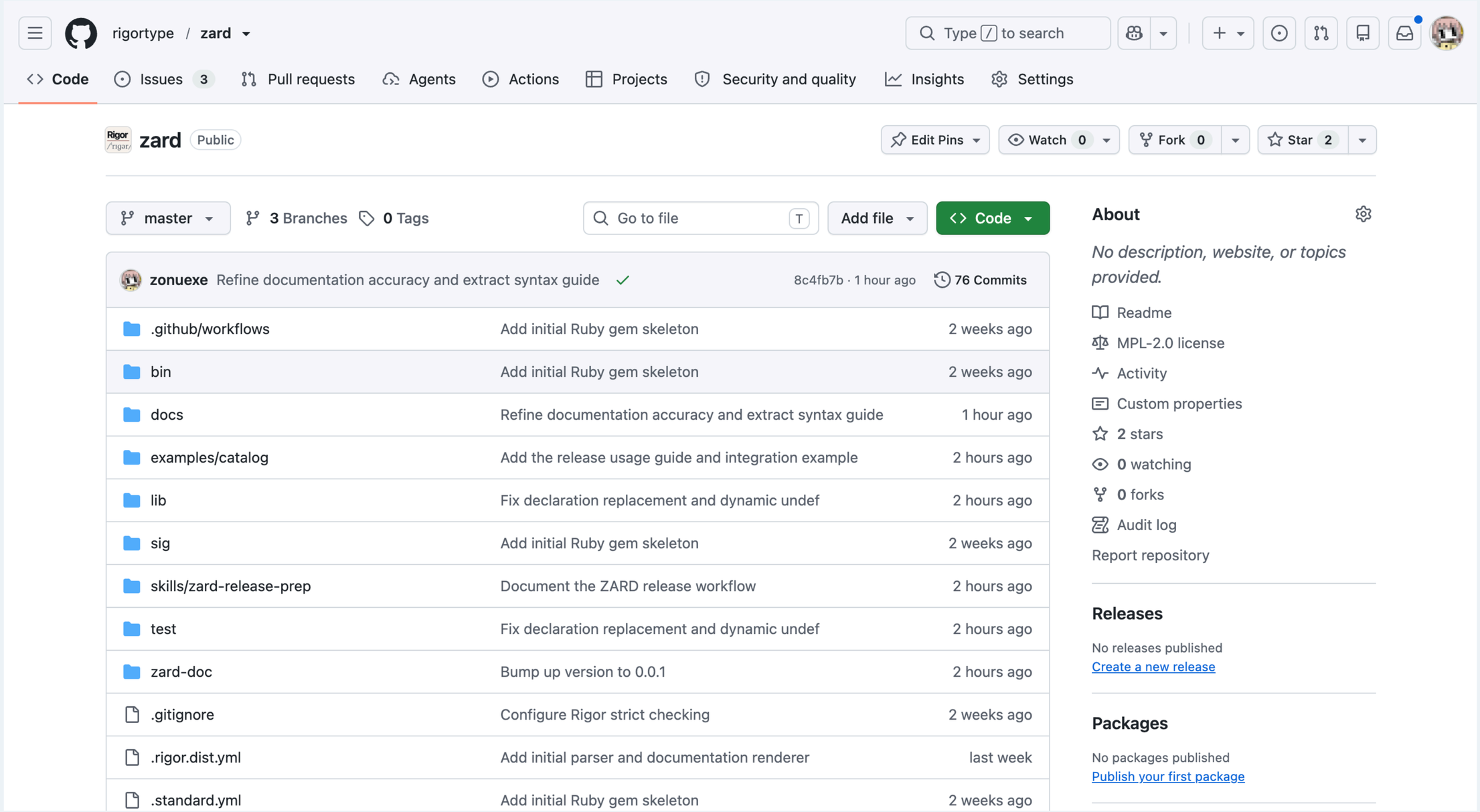
pixiv

なぜrbs-inlineに一本化できないのか

- # @rbs は、あくまで型情報に -- でコメントを載せる記法
  - 主役は型だと認識している
- #: はRBS系エコシステムだけでなくSorbetも読み取るので、
  RBS仕様外の型情報を載せるのはあまりよくない
  - Pythonの型宣言も実行時に読み飛ばされるが、ツールごとの非互換に弱い
- # @extrbs にRigorのRBS::Extendedをコメントに載せられる

作り始めたばかりなので乞うご期待

pixiv


rigortype / zard
Type / to search
Code
Issues 3
Pull requests
Agents
Actions
Projects
Security and quality
Insights
Settings
zard
Public
Edit Pins
Watch 0
Fork 0
Star 2
master
3 Branches
0 Tags
Go to file
T
Add file
Code
zonuexe Refine documentation accuracy and extract syntax guide
8c4fb7b · 1 hour ago
76 Commits
.github/workflows Add initial Ruby gem skeleton 2 weeks ago
bin Add initial Ruby gem skeleton 2 weeks ago
docs Refine documentation accuracy and extract syntax guide 1 hour ago
examples/catalog Add the release usage guide and integration example 2 hours ago
lib Fix declaration replacement and dynamic undef 2 hours ago
sig Add initial Ruby gem skeleton 2 weeks ago
skills/zard-release-prep Document the ZARD release workflow 2 hours ago
test Fix declaration replacement and dynamic undef 2 hours ago
zard-doc Bump up version to 0.0.1 2 hours ago
.gitignore Configure Rigor strict checking 2 weeks ago
.rigor.dist.yml Add initial parser and documentation renderer last week
.standard.yml Add initial Ruby gem skeleton 2 weeks ago
About
No description, website, or topics provided.
Readme
MPL-2.0 license
Activity
Custom properties
2 stars
0 watching
0 forks
Audit log
Report repository
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package

pixiv


Rigor
/ˈrɪgər/



pixiv