

創作プラットフォームにおける コンピュータサイエンス vs 俺たち

Computer Science vs. Us: An Engineering Story from pixiv



pixiv Inc.
USAMI Kenta

pixiv

お前誰よ



- うさみけんた (@tadsan) / Zonu.EXE / にゃんだーすわん
- ■■■工業大学(2007～2012年、現■■■大学)
- 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
- Emacs PHP Modeを開発しています(2017年-)
- プログラミング言語にこだわりがある(セキュリティ&プログラミング2010)

お前誰よ



- うさみけんた (@tadsan) / Zonu.EXE / にゃんだーすわん
- 北海道工業大学(2007～2012年、現 大学)
- 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
- Emacs PHP Modeを開発しています(2017年-)
- プログラミング言語にこだわりがある(セキュリティ&プログラミング2010)

お前誰よ



- うさみけんた (@tadsan) / Zonu.EXE / にゃんだーすわん
- 北海道工業大学(2007～2012年、現北海道科学大学)
- 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
- Emacs PHP Modeを開発しています(2017年-)
- プログラミング言語にこだわりがある(セキュリティ&プログラミング2010)

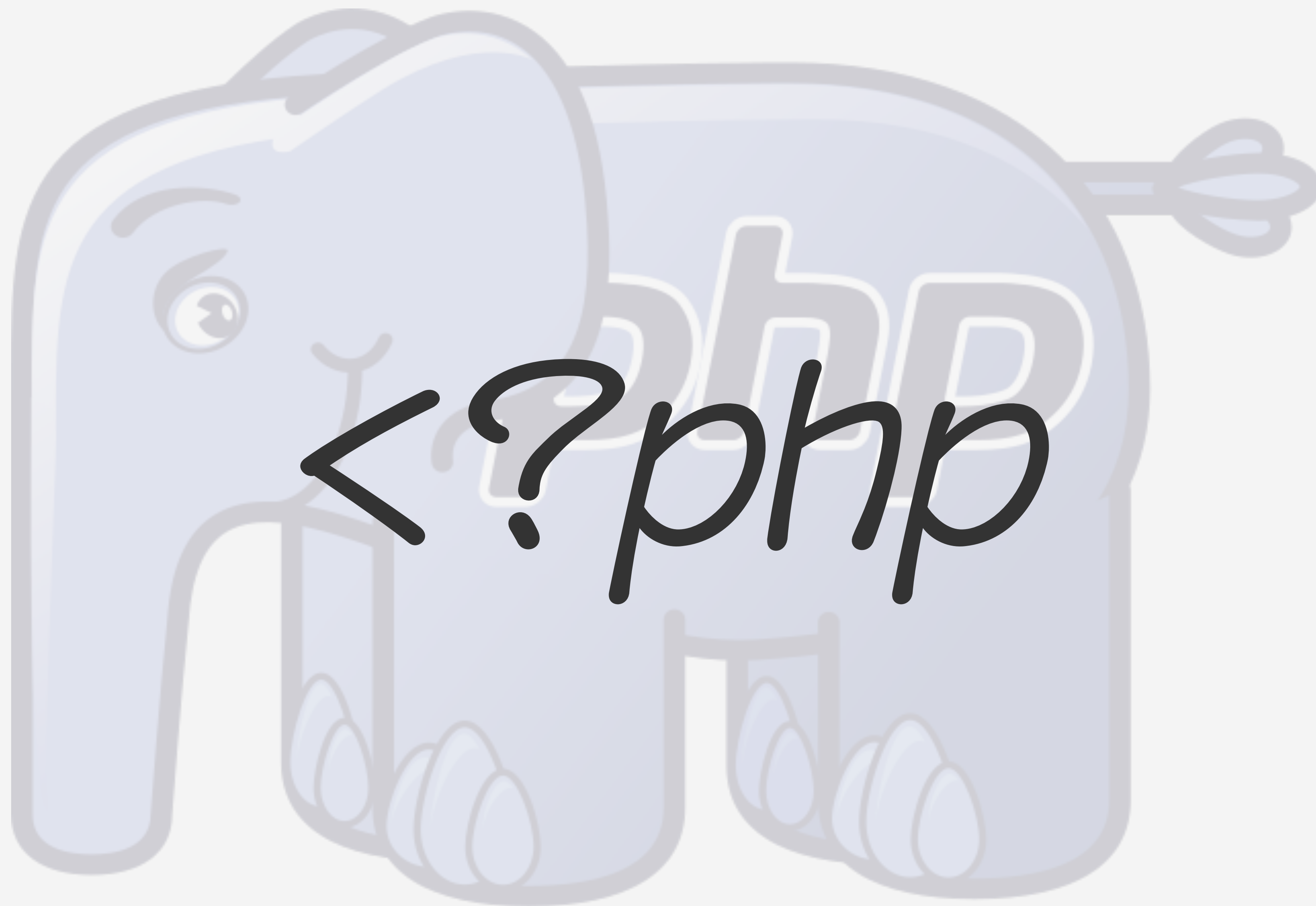
お前誰よ

CSっぽいカリキュラム
全然なくて絶望した



- うさみけんた (@tadsan) / Zonu.EXE / にゃんだーすわん
- 北海道工業大学(2007～2012年、現北海道科学大学)
- 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
- Emacs PHP Modeを開発しています(2017年-)
- プログラミング言語にこだわりがある(セキュリティ&プログラミング2010)

普段やってる仕事



pixiv

PIXIVコミック

BOOTH

pixiv FACTORY

PIXIV FANBOX

FANBOXプリント

ImageFlux
Powered by pixiv

SKETCH

PixiVision

sensei

ピクエ百科事典

Palcy パルシィ

pixiv comic
indies

VRoid Studio

VRoid Hub

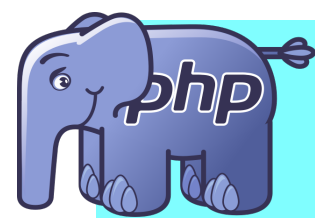
pixiv Ads

pixiv WAENGALLERY
by TWIN PLANET × pixiv

pixiv coban

Pastela

beat

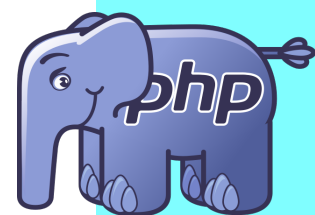


pixiv

PIXIVコミック

BOOTH

pixiv FACTORY

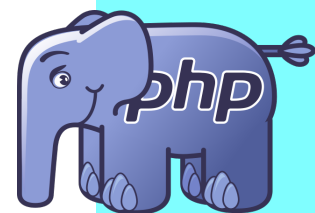


PIXIV FANBOX

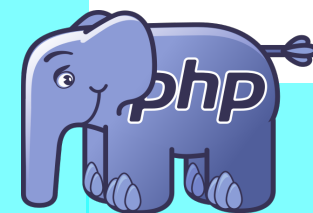
FANBOXプリント

ImageFlux
Powered by pixiv

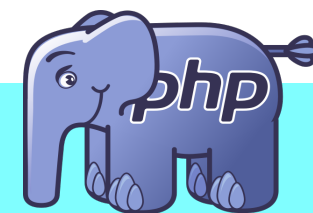
SKETCH



PixiVision

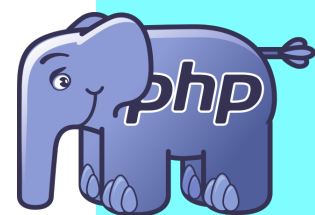


sensei



ピクエ百科事典

Palcy
パルシィ



pixiv comic
indies

VRoid Studio

VRoid Hub

pixiv Ads

pixiv WAENGALLERY
by TWIN PLANET × pixiv

pixiv coban

Pastela

beat
ビート



 pixiv coban

Pastela



pixivとDBを直接
共有しているのは
PHP (モノレポ)

それ以外は
開発時期や体制に
応じていろいろ

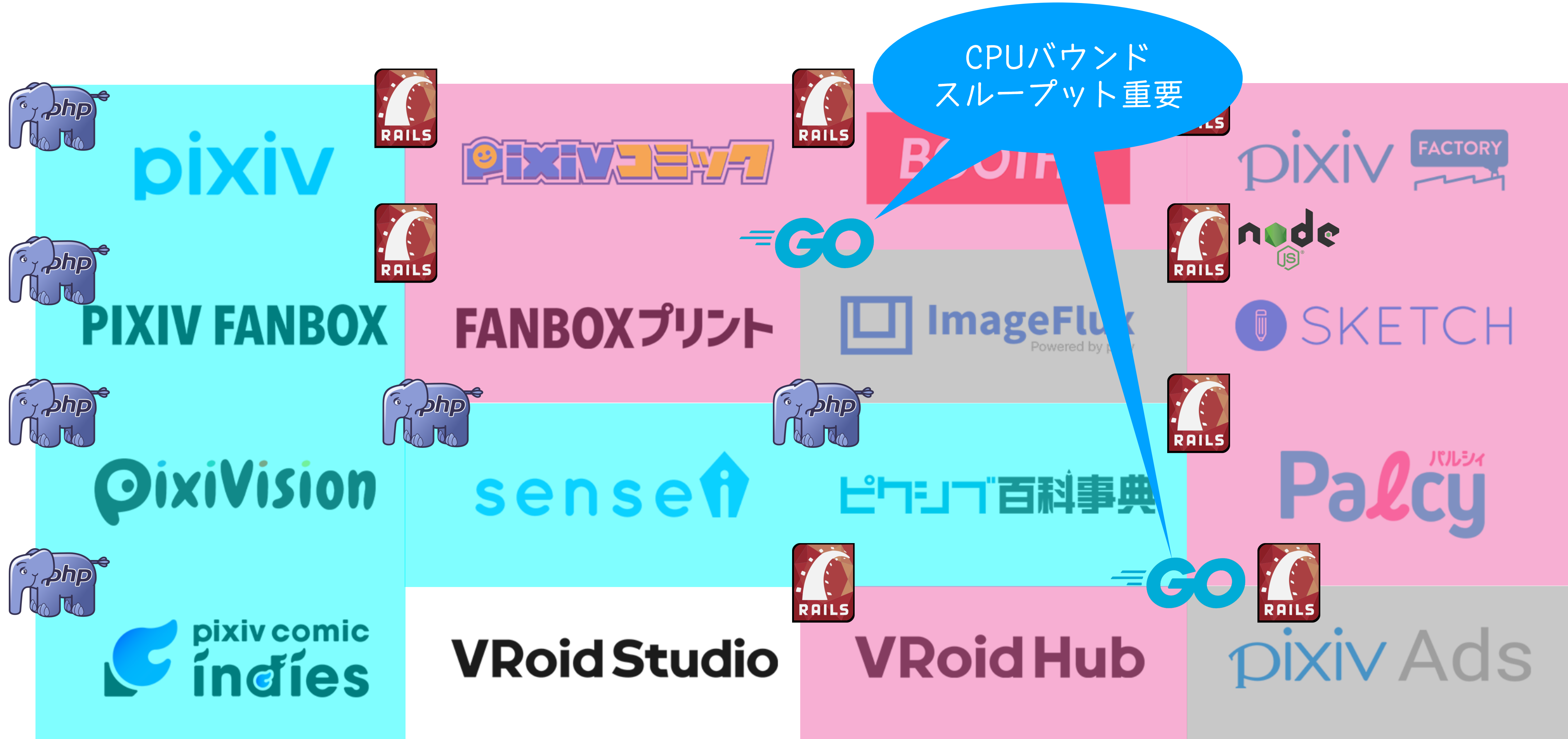


pixivWAENGALLERY
by TWIN PLANET × pixiv

 pixiv coban

Pastela



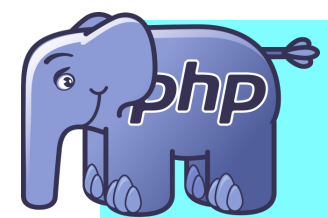


pixivWAENGALLERY
by TWIN PLANET × pixiv

 pixiv coban

Pastela

 veat ^{ビート}



pixiv

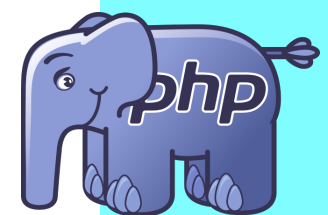


PIXIVコミック



CPUバウンド
スループット重要

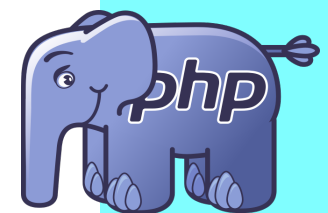
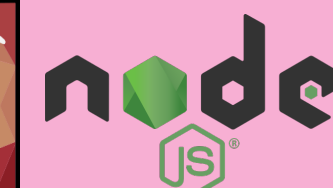
WebSocket
C10K問題



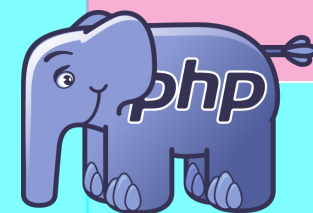
PIXIV FANBOX



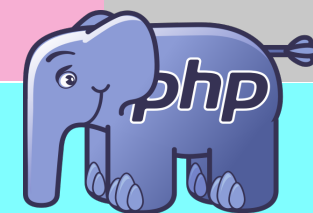
FANBOXプリント



PixiVision



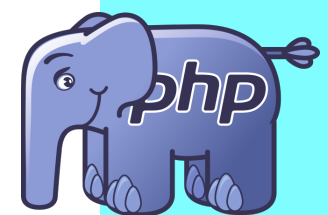
sensei



ピクエ百科事典



SKETCH



pixiv comic
indies



VRoid Studio



VRoid Hub

pixiv Ads

Palcy
パルシィ

pixivWAENGALLERY
by TWIN PLANET × pixiv

pixiv coban

Pastela

veat
ビート



PHP: Hypertext Preprocessor

- Web開発にしか使えない、遅い動的言語… と考えられている
- 型がない、あるいは貧弱な言語… と、認識されている
 - ちょっとぴりC言語っぽい標準関数と->
 - そこはかたなくJavaっぽいオブジェクト指向
 - とてもPerlっぽい構文（特に変数の \$ と、文末の ; ）

PHP: Hypertext Processor

こういう言語の
型解析とかFWについて
発信してます



- Web開発にしか使えない、遅い動的言語… と考えられている
- 型がない、あるいは貧弱な言語… と、認識されている
 - ちょっぴりC言語っぽい標準関数と->
 - そこはかたなくJavaっぽいオブジェクト指向
 - とてもPerlっぽい構文（特に変数の \$ と、文末の ; ）

PHPは見る人の心を映す



現役の学生には
PHPへの感情は
特にならないかも



20年以上Webを
支えている言語

個人的には
ハックしがいのある
おもしろ言語



技術職向け夏インターンシップ『PIXIV SUMMER BOOT CAMP 2025』 基盤/プラットフォームコース

インターン

ピクシブ株式会社 の求人一覧

✕ ポスト

シェアする 0

Share





技術職向け夏インターンシップ『PIXIV SUMMER BOOT CAMP 2025』 基盤/プラットフォームコース

インターン

ピクシブ株式会社 の求人一覧

✕ ポスト

シェアする 0

Share

PIXIV SUMMER BOOT CAMP

基盤 / プラットフォーム

プロダクトの根幹を担う技術の基盤に携われるコース

■技術的挑戦

pixivでは静的解析を単純なCIや開発時のLintとして活用しているだけでなく、拡張機能によって標準機能では実現できない型付けをしたり、Rectorというリファクタリングツールの独自のルールを運用しています。メンターのtadsanは静的解析ツールやRectorの社内ルールの開発・メンテナンスのほか、必要に応じてupstreamへの貢献も行っています。

■コースでできる体験

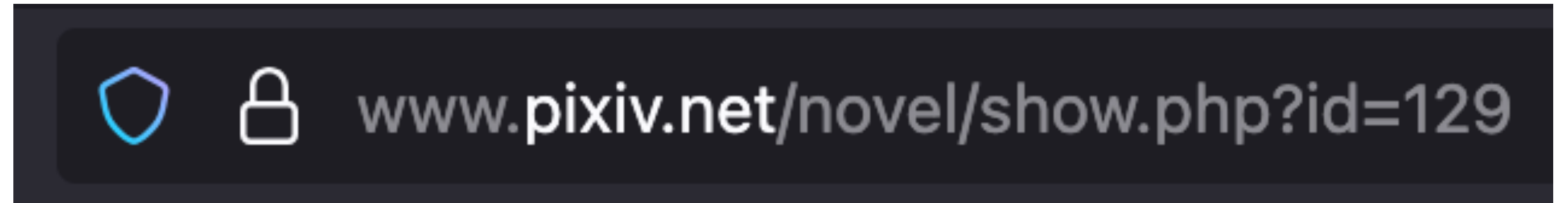
静的解析ツールまたはRectorを用いて、型システムおよび構文木に基いた静的解析に挑戦します。特定のプログラミング言語だけでなく、プログラミング言語そのものに強い興味がある方を歓迎します。PHPの経験はあったと望ましいですが、必須ではありません。

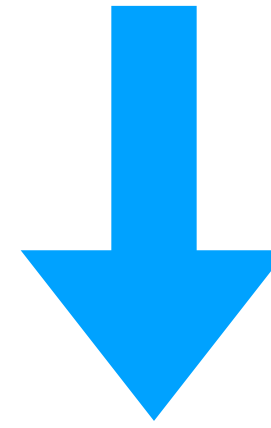
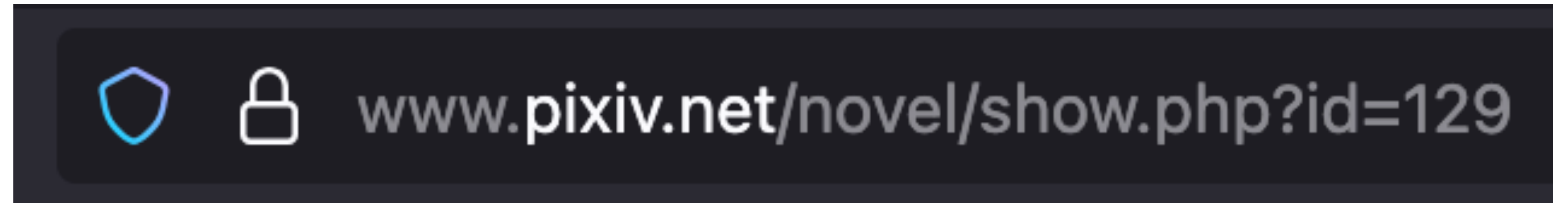
pixivは2007年から
絶え間なく
開発が続いている

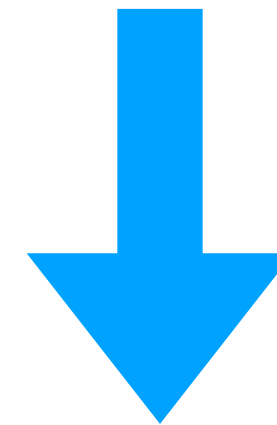
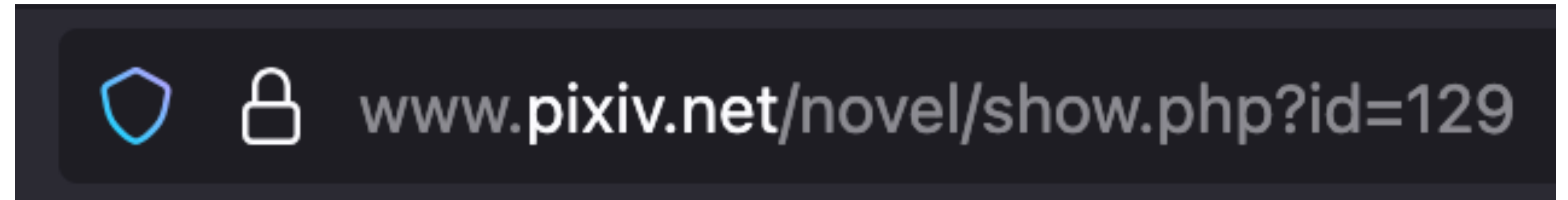
いろいろな人たちの
問題解決が
詰まっている

さも自分がやった
仕事のように紹介します

URLルーティング vs 俺たち

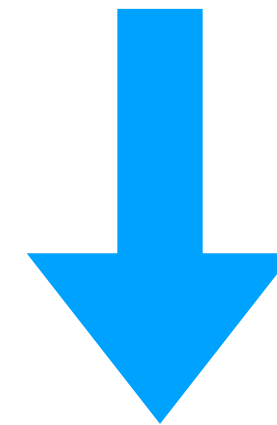
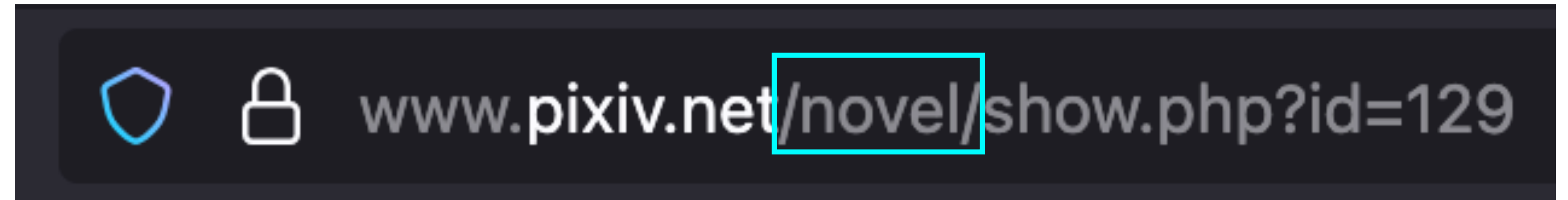






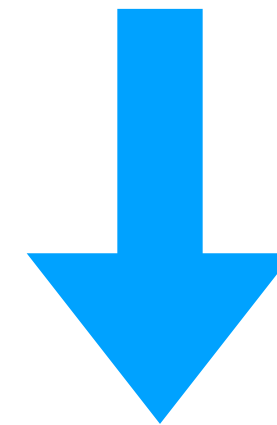
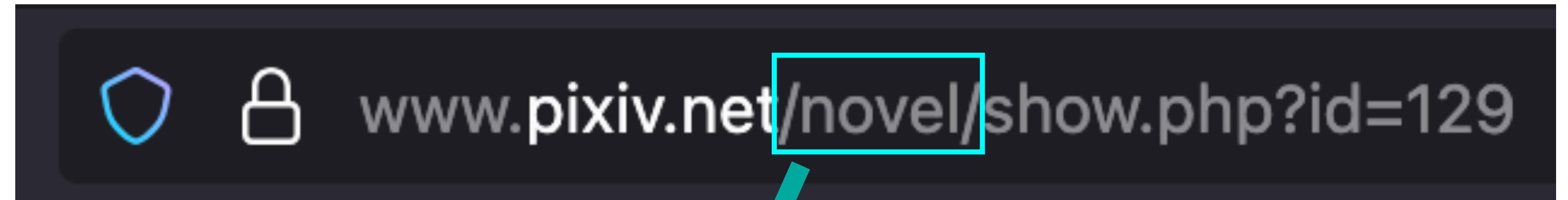
/data/www/www.pixiv.net (document root)

- index.php
- novel
 - show.php
- yyy.php
- zzz.php



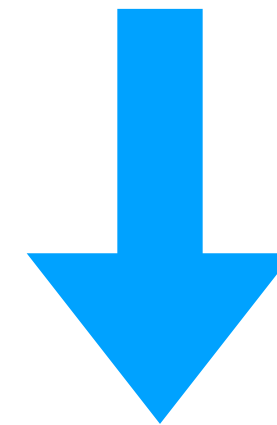
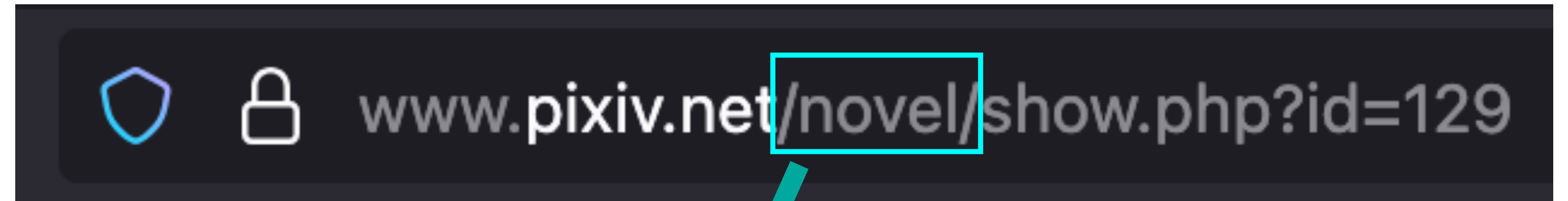
/data/www/www.pixiv.net (document root)

- index.php
- novel
 - show.php
- yyy.php
- zzz.php



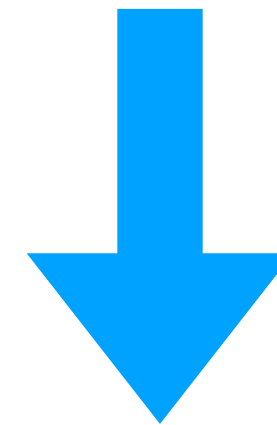
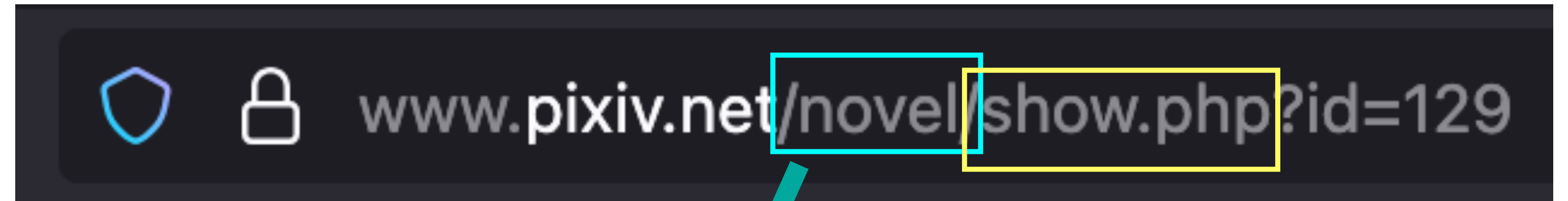
/data/www/www.pixiv.net (document root)

- index.php
- novel
 - show.php
- yyy.php
- zzz.php



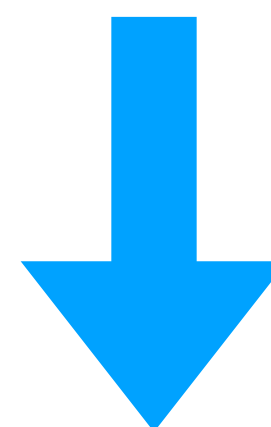
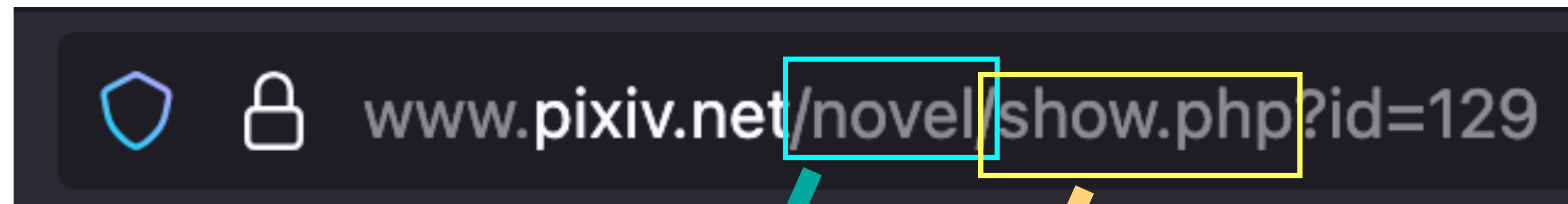
/data/www/www.pixiv.net (document root)

- index.php
- **novel**
 - show.php
- yyy.php
- zzz.php



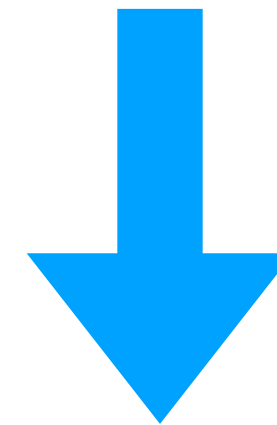
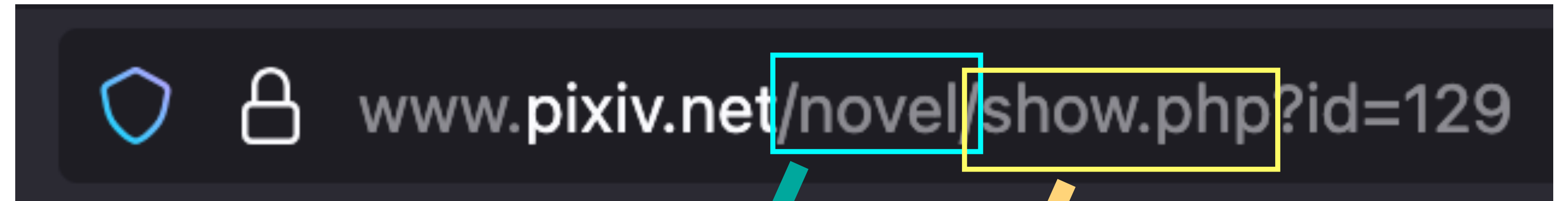
/data/www/www.pixiv.net (document root)

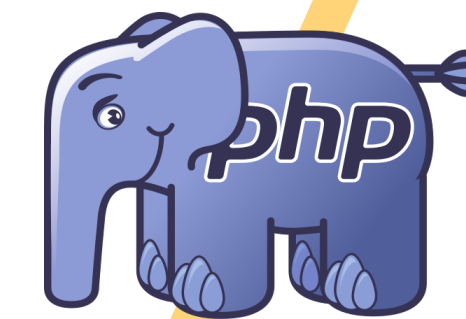
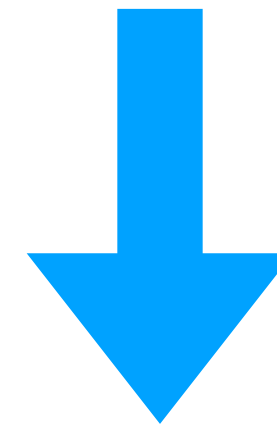
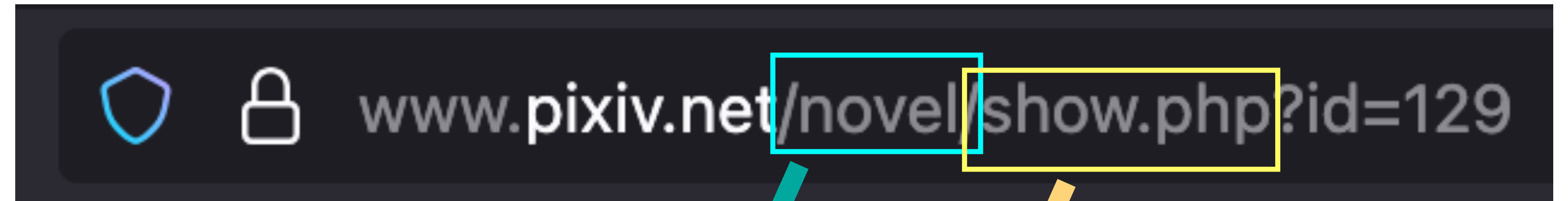
- index.php
- novel
 - show.php
- yyy.php
- zzz.php

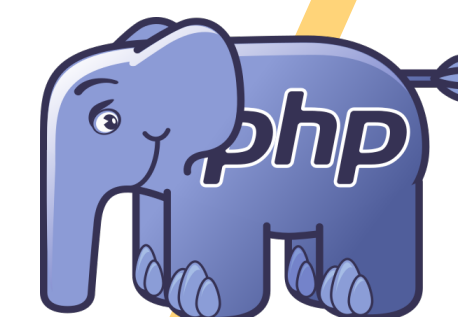
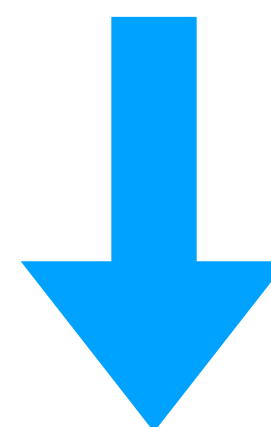
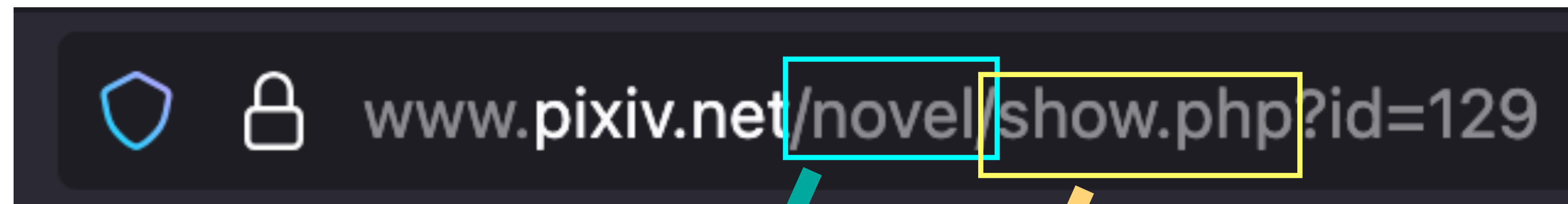


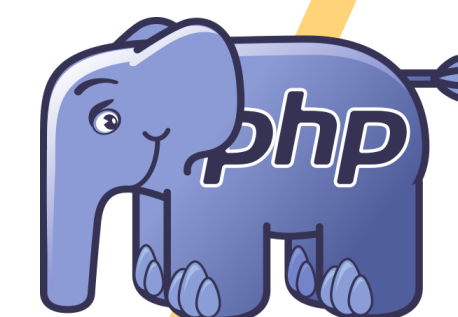
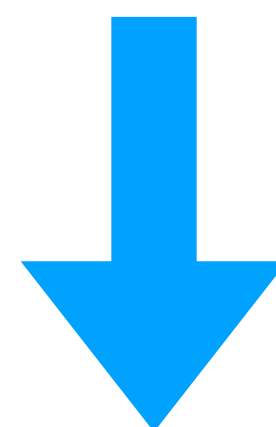
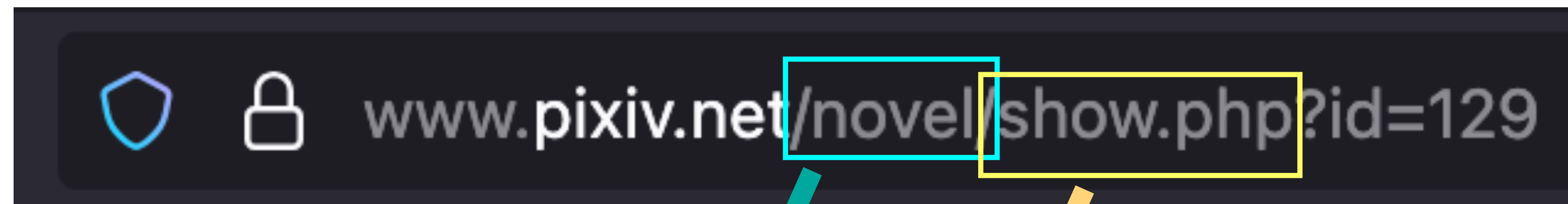
/data/www/www.pixiv.net (document root)

- index.php
- novel
 - show.php
- yyy.php
- zzz.php



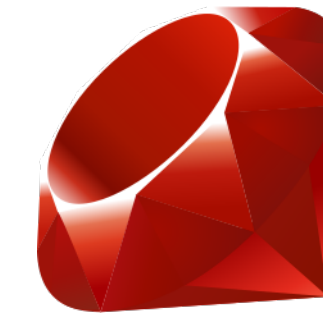
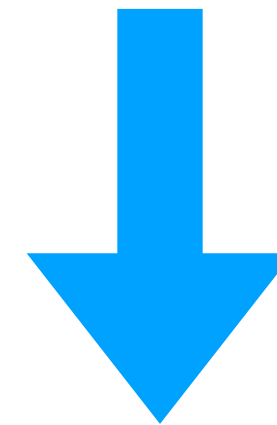
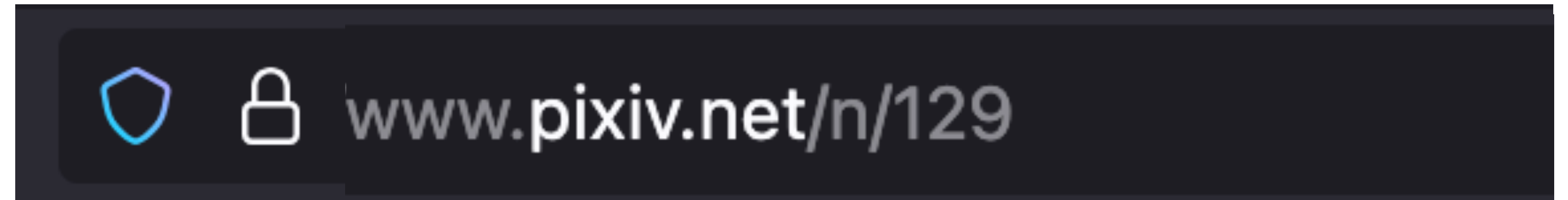






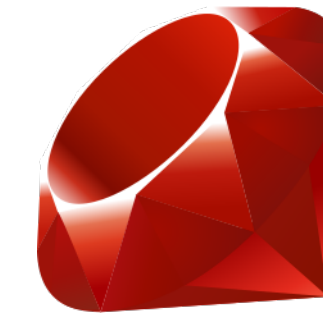
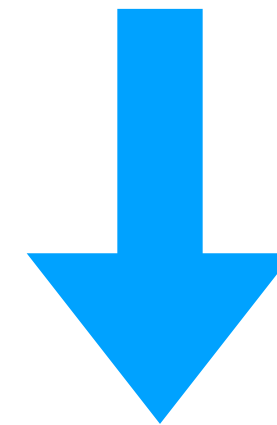
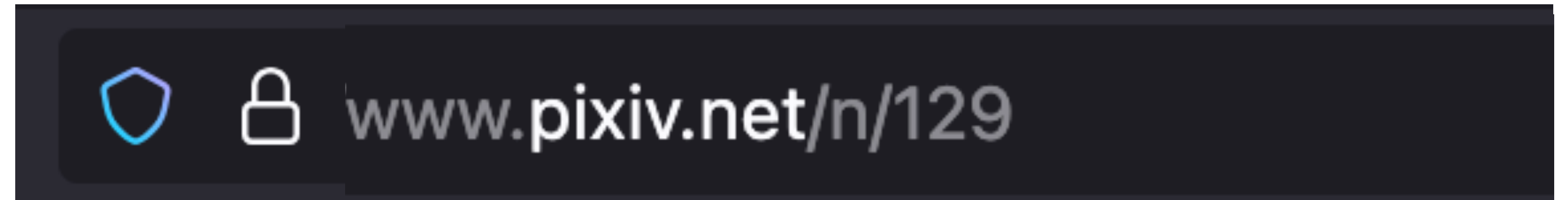
ファイルをサーバに
コピーすれば
デプロイ完了する

Railsとか使えば
綺麗なURLになるよね



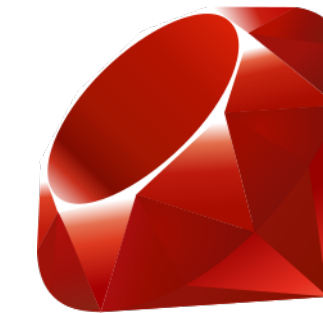
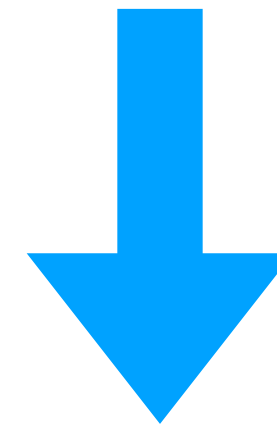
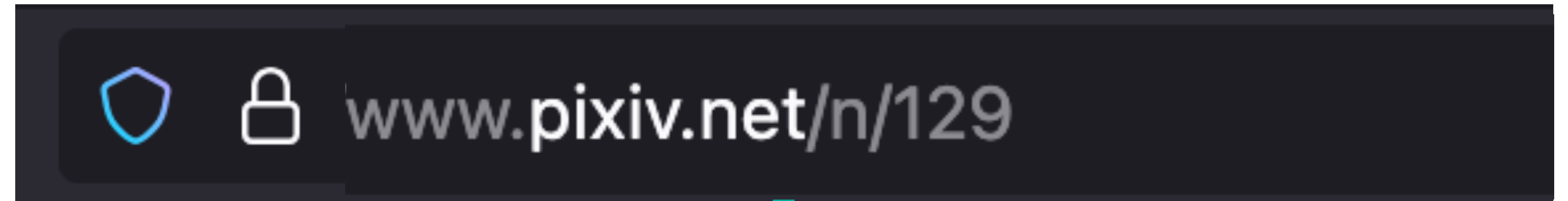
NovelController#show

```
# routes.rb
get '/', to: 'index'
get '/n/:id', to: 'novel#show'
get '/u/:id', to: 'user#show'
get '/yyy', to: 'yyy'
get '/zzz', to: 'zzz'
```



NovelController#show

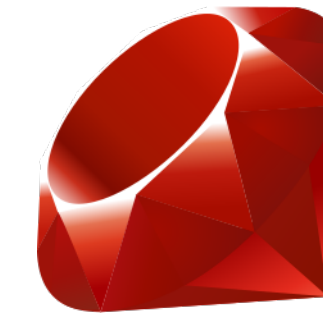
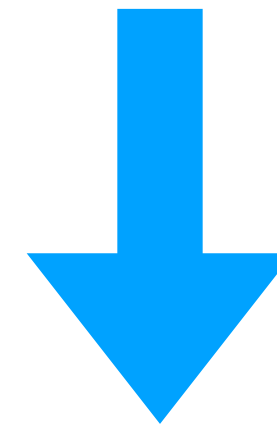
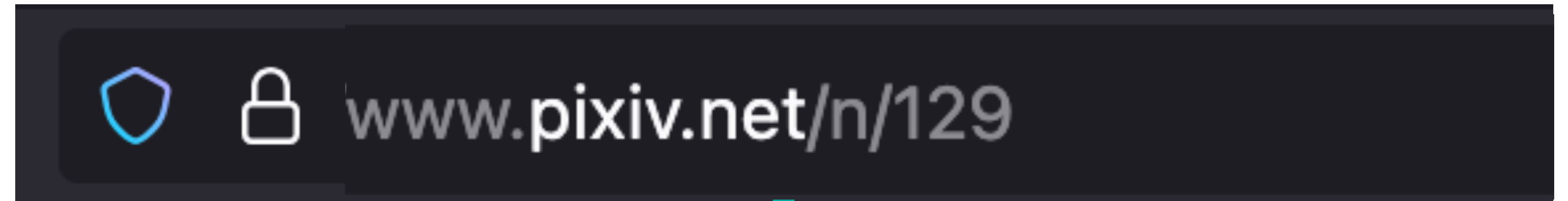
```
# routes.rb
get '/', to: 'index'
get '/n/:id', to: 'novel#show'
get '/u/:id', to: 'user#show'
get '/yyy', to: 'yyy'
get '/zzz', to: 'zzz'
```



NovelController#show



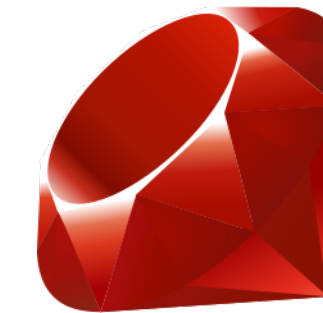
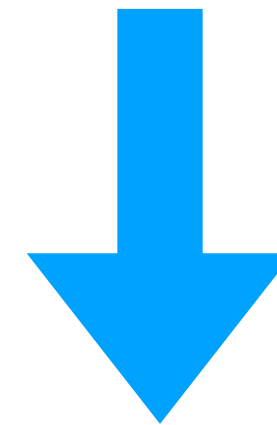
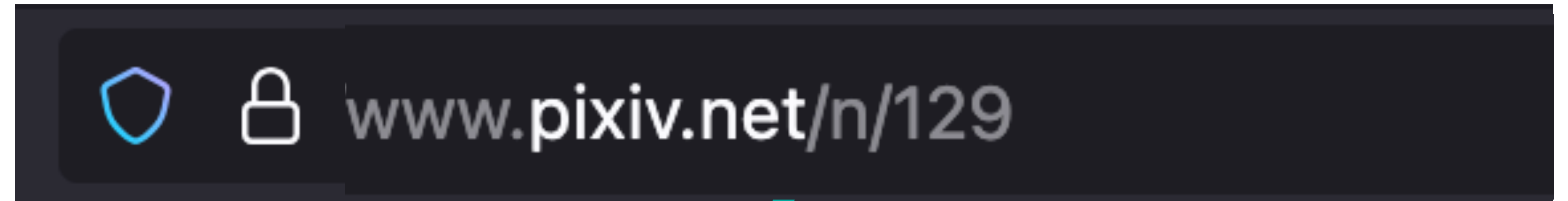
```
# routes.rb
get '/', to: 'index'
get '/n/:id', to: 'novel#show'
get '/u/:id', to: 'user#show'
get '/yyy', to: 'yyy'
get '/zzz', to: 'zzz'
```



NovelController#show



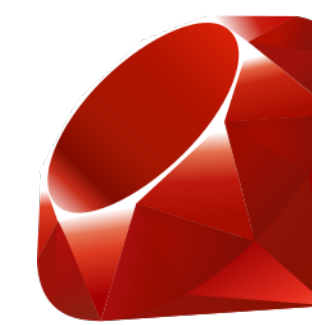
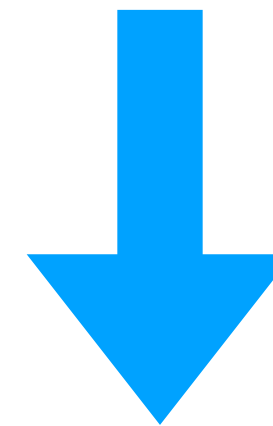
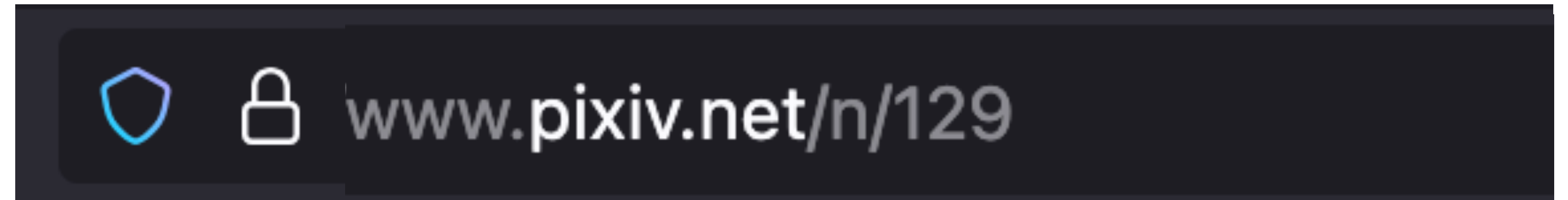
```
# routes.rb
get '/', to: 'index'
get '/n/:id', to: 'novel#show'
get '/u/:id', to: 'user#show'
get '/yyy', to: 'yyy'
get '/zzz', to: 'zzz'
```



NovelController#show



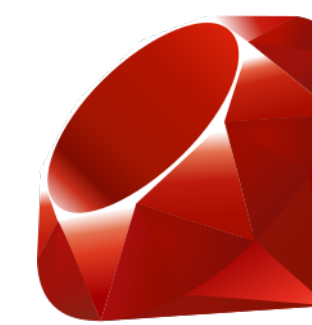
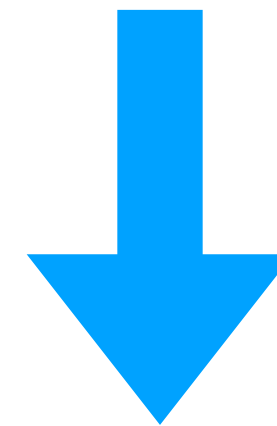
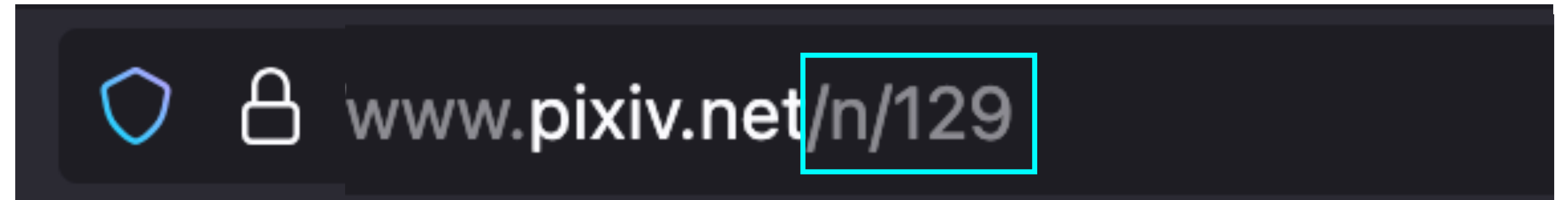
```
# routes.rb  
get '/', to: 'index'  
get '/n/:id', to: 'novel#show'  
get '/u/:id', to: 'user#show'  
get '/yyy', to: 'yyy'  
get '/zzz', to: 'zzz'
```



NovelController#show



```
# routes.rb
get '/', to: 'index'
get '/n/:id', to: 'novel#show'
get '/u/:id', to: 'user#show'
get '/yyy', to: 'yyy'
get '/zzz', to: 'zzz'
```



NovelController#show



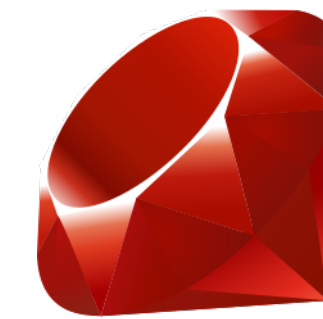
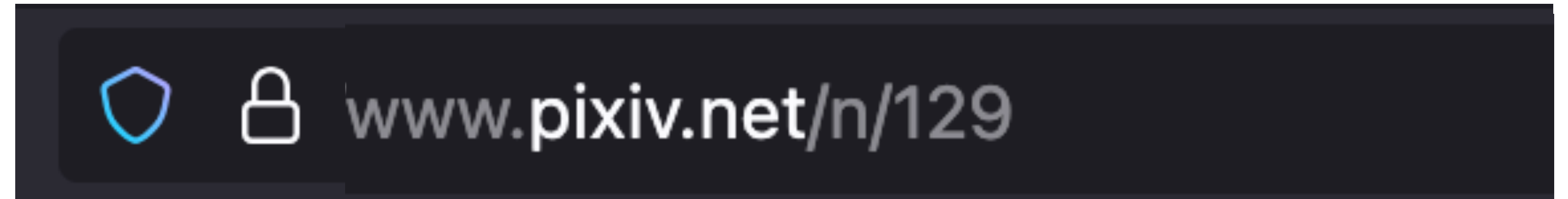
```
# routes.rb
get '/', to: 'index'
get '/n/:id', to: 'novel#show'
get '/u/:id', to: 'user#show'
get '/yyy', to: 'yyy'
get '/zzz', to: 'zzz'
```



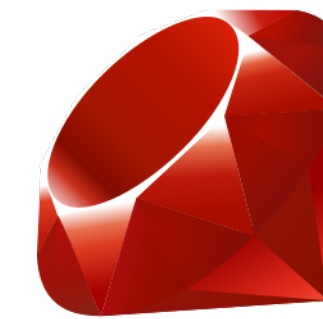
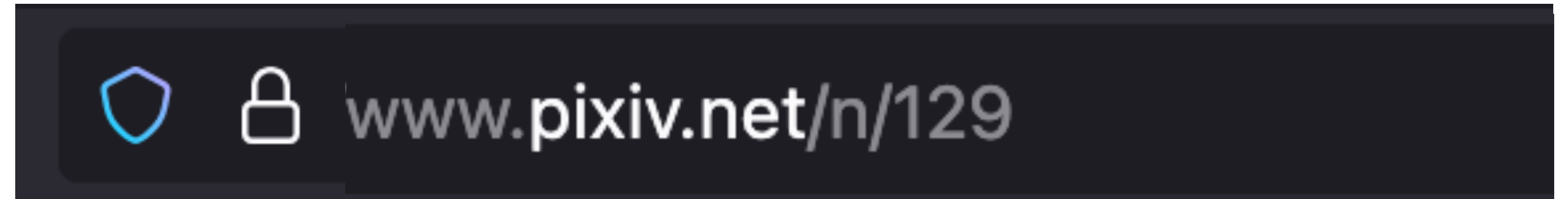
pixivは
フレームワークに
乗ってない

時は流れて...

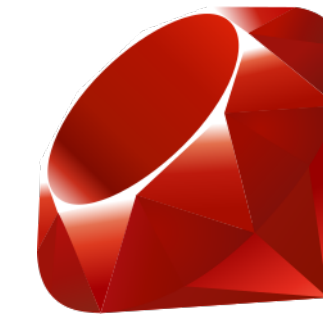
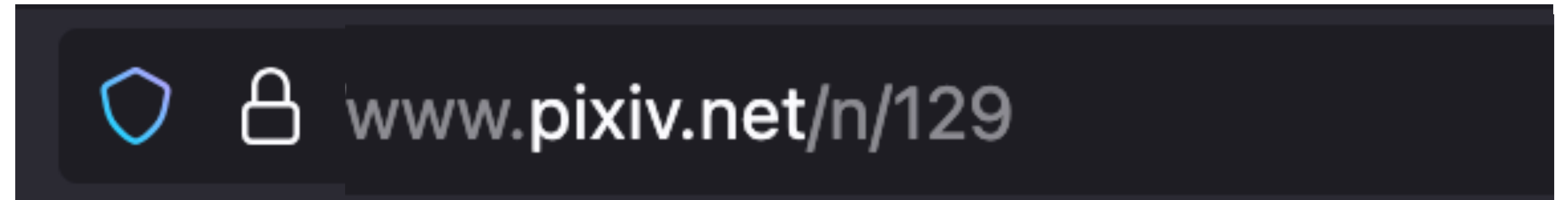
pixiv本体にも URL正規化の流れ



```
# routes.rb
get '/', to: 'index'
get '/n/:id', to: 'novel#show'
get '/u/:id', to: 'user#show'
get '/yyy', to: 'yyy'
get '/zzz', to: 'zzz'
```

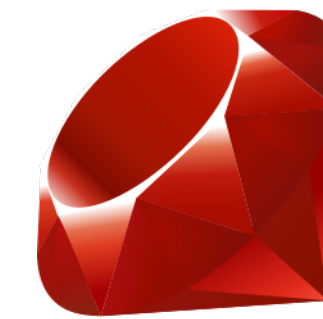
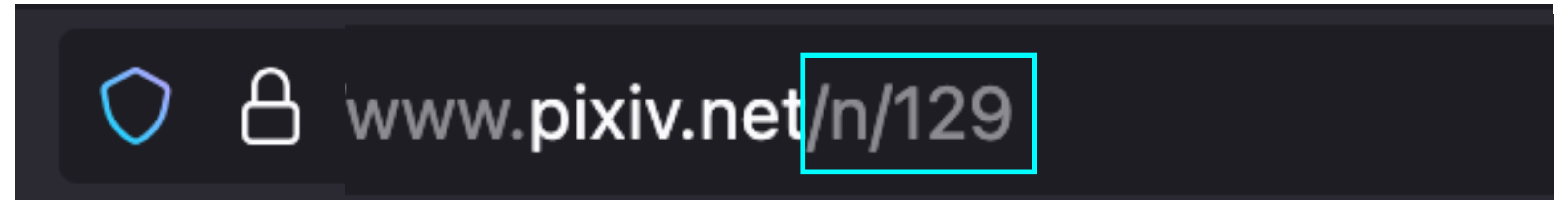


```
# routes.rb  
get '/', to: 'index'  
get '/n/:id', to: 'novel#show'  
get '/u/:id', to: 'user#show'  
get '/yyy', to: 'yyy'  
get '/zzz', to: 'zzz'
```



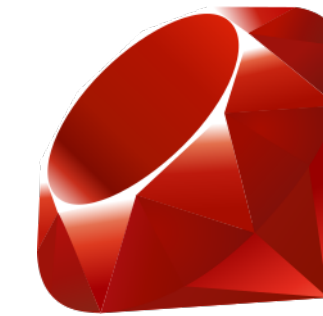
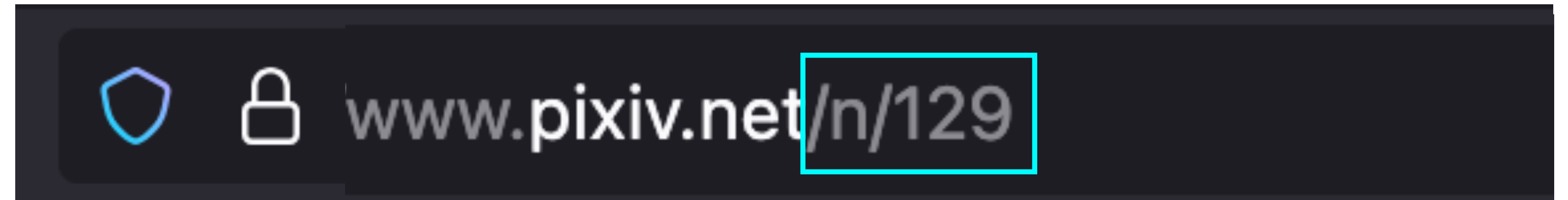
```
# routes.rb  
get '/', to: 'index'  
get '/n/:id', to: 'novel#show'  
get '/u/:id', to: 'user#show'  
get '/yyy', to: 'yyy'  
get '/zzz', to: 'zzz'
```

定数URLは
ハッシュテーブルで
 $O(n)$ で取得



```
# routes.rb  
get '/', to: 'index'  
get '/n/:id', to: 'novel#show'  
get '/u/:id', to: 'user#show'  
get '/yyy', to: 'yyy'  
get '/zzz', to: 'zzz'
```

定数URLは
ハッシュテーブルで
 $O(n)$ で取得



URLにIDが
組み込まれている場合は…?

定数URLは
ハッシュテーブルで
 $O(n)$ で取得

```
# routes.rb
get '/', to: 'index'
get '/n/:id', to: 'novel#show'
get '/u/:id', to: 'user#show'
get '/yyy', to: 'yyy'
get '/zzz', to: 'zzz'
```

ボトルネックじゃなけりゃ
雑に処理すればいいじゃん
(大富豪の発想)

シンプルなルーティングがしたかった

PHP

最終更新日 2018年06月26日 投稿日 2015年03月06日

Slimとかあるけど、僕は別に好きじゃないんだ。

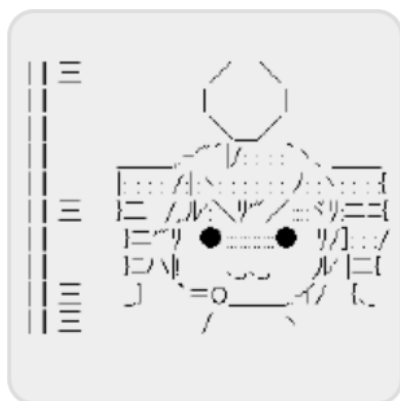
はじめに

要はWebアプリってHTTPリクエストから任意の関数を起動できりゃいいのだけれども、関数の引数は単なるデータに過ぎないわけで、ルーターの仕事はメソッドとPathから固有の値を返してくりそれだけでいい。

そんなこんなのでTeto Routingを作った。以下のようなコマンドでインストールできる。

```
composer require zonuxe/simple-routing
```

ルーターをプロジェクトに組み込む方法は[インスパイアされて掲示板を作りたくなった\(3\)](#)に書いた。



Baguette HQ

ξ ^ω^)ξ Kawaii PHP Libraries

1 follower https://baguettephp.github.io/ tadsan@pixiv.com

Pinned

Customize pins

TetoRouting Public
Simple routing for WebApp
● PHP ☆ 21 🍴 2

objectsystem Public
Object system implementation for PHP
● PHP ☆ 6

URLルーター作った！
(ここまでは自分の仕事)

pixiv本体にも URL正規化の流れ

Teto Routingは、**実行時間がルーティング数に依存する** 実装になっています。ここでいう実行時間とは、Teto RoutingにリクエストURL文字列を渡してから結果が返ってくるまでの時間のことです。実際、Teto Routingは表Iのような時間がかかります。私の環境なので数値自体にあまり意味はなく、ルーティング数に線形比例して実行時間が増えているのが注目ポイントです。

表I. Teto Routingの実行時間

ルーティング数	実行時間 (ms)
5	0.328
50	2.515
100	4.995

Teto Routingは、**実行時間がルーティング数に依存する** 実装になっています。ここでいう実行時間とは、Teto RoutingにリクエストURL文字列を渡してから結果が返ってくるまでの時間のことです。実際、Teto Routingは表Iのような時間がかかります。私の環境なので数値自体にあまり意味はなく、ルーティング数に線形比例して実行時間が増えているのが注目ポイントです。

表I. Teto Routingの実行時間

ルーティング数	実行時間 (ms)
5	0.328
50	2.515
100	4.995

たったの100個でこの有様
これだから線形時間はだめ

どうにかしよう
(と競プロ得意な同僚が言った)

URLルーティングには
実装の定石がある

正規表現ベース実装

- 全部のルーティング対象のパスを結合したパターンを構築して、リクエストされたパスをマッチさせる
- めちゃくちゃ力技っぽく見えるが… アイディアはシンプル
- ところで… 現代の実用的な正規表現エンジンはJITコンパイルされ、速い

基数木(パトリシアトライ)

- パスを“/”で区切りってネストした連想配列を構築する

基数木(パトリシアトライ)

- パスを“/”で区切りってネストした連想配列を構築する

```
get '/', to: 'index'  
get '/xxx', to: 'xxx#index'  
get '/xxx/:id', to: 'xxx#shoq'  
get '/xxx/yyy', to: 'yyy'  
get '/xxx/zzz', to: 'zzz'
```

基数木(パトリシアトライ)

- パスを“/”で区切りってネストした連想配列を構築する

```
get '/', to: 'index'  
get '/xxx', to: 'xxx#index'  
get '/xxx/:id', to: 'xxx#show'  
get '/xxx/yyy', to: 'yyy'  
get '/xxx/zzz', to: 'zzz'
```

```
{  
  "", : {"#result": "index"}  
  "xxx", : {  
    "": {"#result": "xxx#index"},  
    ":id": {"#result": "xxx#show"},  
    "yyy": {"#result": "yyy"},  
    "zzz": {"#result": "zzz"},  
    ...  
  }  
}
```

では作ろう

(さも自分がやったかのように)

では作ろう
(さも自分がやったかのように)

正規表現 VS パトリシアトライ

pixiv inside [archive]



pixiv insideは移転しました！ » <https://inside.pixiv.blog/>

2015-12-13

PHPで高速に動作するURLルーティングを自作してみた

164

B!ブックマーク

0

シェアする

この記事は ピクシブ株式会社 Advent Calendar 2015 13日目の記事です。

ピクシブ株式会社 - Qiita Advent Calendar 2015 - Qiita

Calendar page for Qiita Advent Calendar 2015 regarding ピクシブ株式会社.

qiita.com **32 users**



pixiv insideとは

<https://inside.pixiv.blog/> に移転しました。こちらは2016年末までのアーカイブです。

✓ 読者です 237

[このブログについて](#)

月別アーカイブ

- ▼ 2016 (50)
- 2016 / 12 (27)
- 2016 / 11 (6)
- 2016 / 10 (1)
- 2016 / 9 (3)
- 2016 / 8 (1)
- 2016 / 7 (2)
- 2016 / 6 (1)

選ばれたのは
パトリシアトライ
でした

パトリシアトライを実装しよう

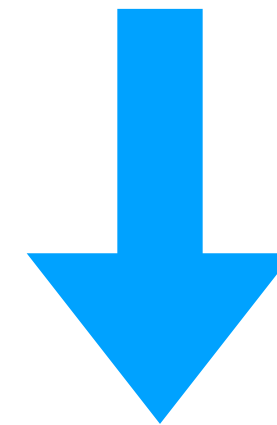
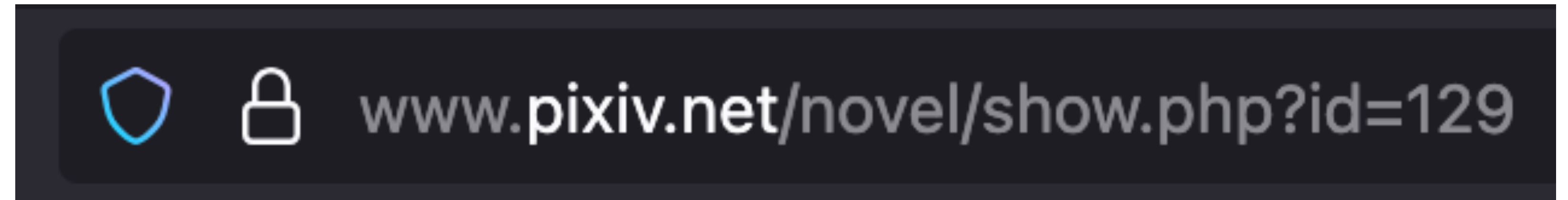
- 正規表現に比べアルゴリズムが素朴で、実装もめっちゃくちゃ簡単
 - パスのリストから “/” でネストした多次元の連想配列に変換するだけ
 - 可変パラメータは数値と文字列のみ対応（複雑なパターンも対応は可能）
- 構築コストはルーティング数によって線形増加するが、十分許容できる
 - キャッシュも可能だが、いまのところはボトルネックになっていない
- マッチしない場合も処理時間のペナルティがない

ルーティングは
できた

めでたしめでたし

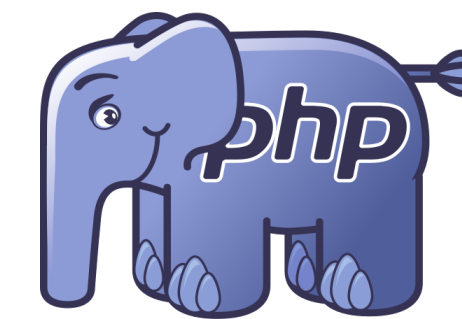
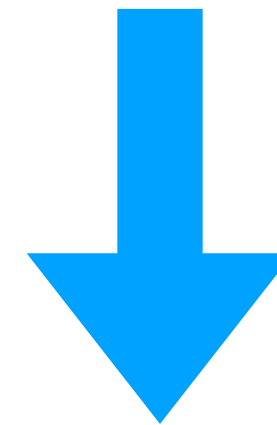
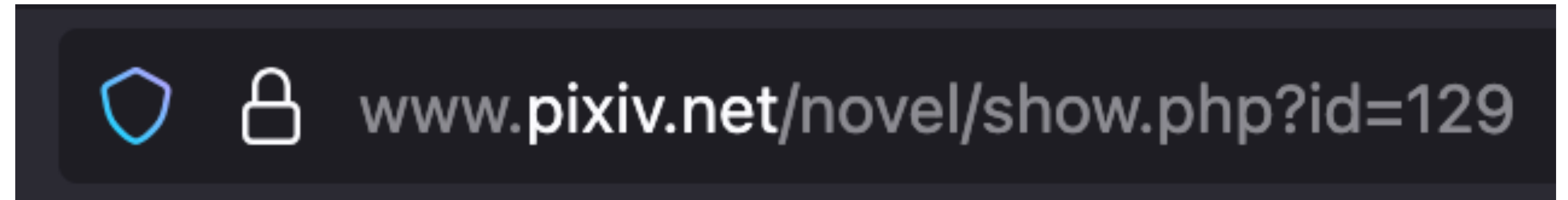


あとは既存コードを
URLルーターに
載せればいいだけ



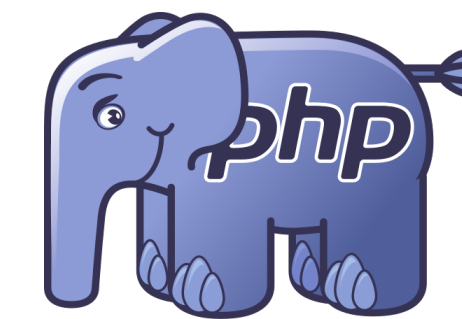
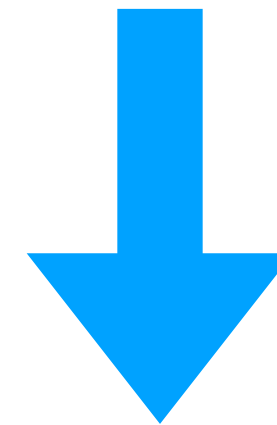
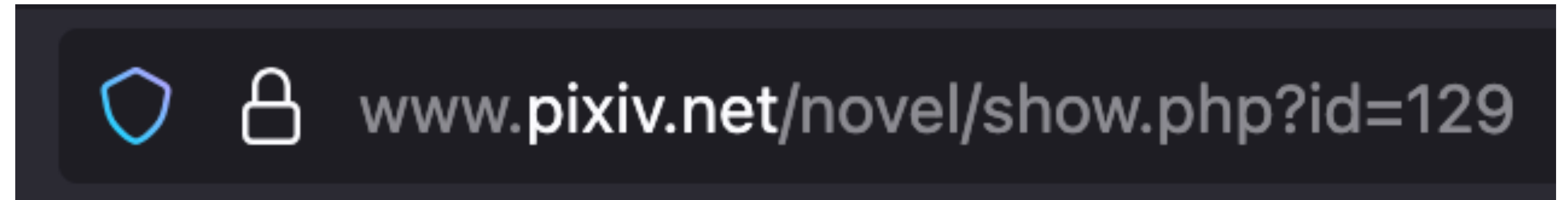
/data/www/www.pixiv.net (document root)

- index.php
- novel
 - show.php
- yyy.php
- zzz.php



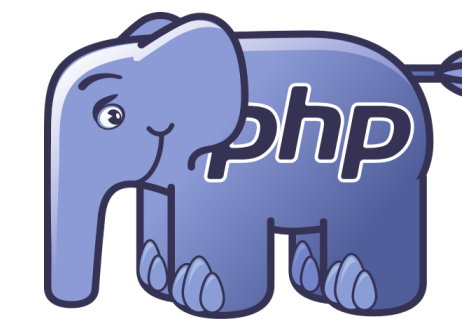
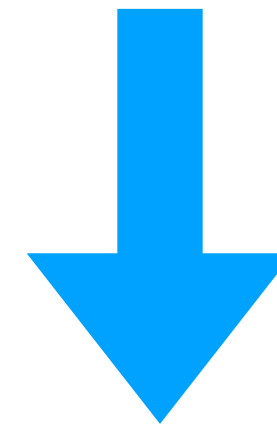
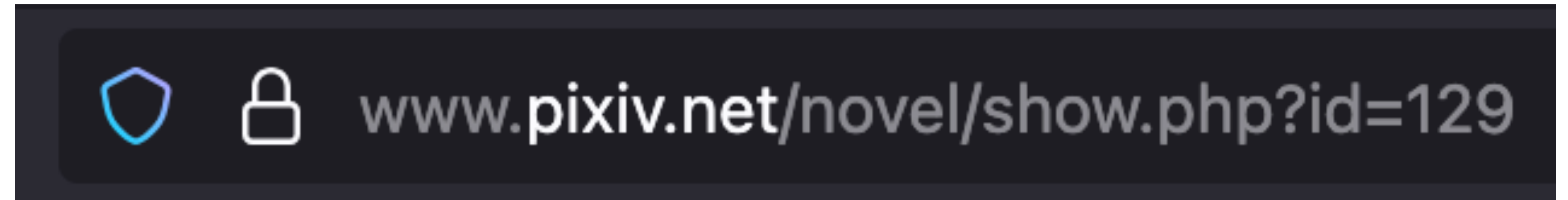
/data/www/www.pixiv.net (document root)

- index.php
- novel
 - show.php
- yyy.php
- zzz.php



/data/www/www.pixiv.net (document root)

- index.php
- novel
 - show.php
- yyy.php
- zzz.php



/data/www/www.pixiv.net (document root)

- index.php
- novel
 - show.php
- yyy.php
- zzz.php

現実にはファイルが
何百個もある！！！！



/data/www/www.pixiv.net

├── index.php

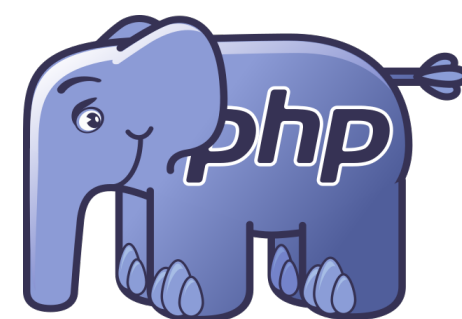
├── novel

│ └── show.php

├── yyy.php

└── zzz.php

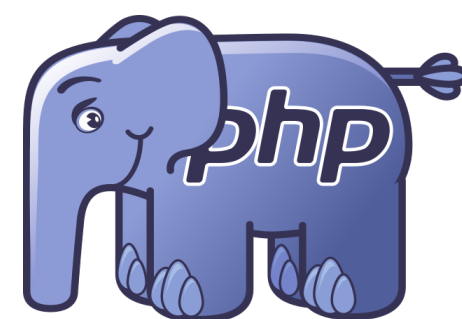
(実際にはファイルが数百個)



/data/www/www.pixiv.net

- |—— index.php
- |—— novel
 - |—— show.php
- |—— yyy.php
- |—— zzz.php

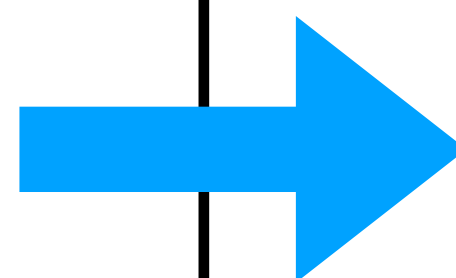
(実際にはファイルが数百個)



/data/www/www.pixiv.net

- |—— index.php
- |—— novel
 - |—— show.php
- |—— yyy.php
- |—— zzz.php

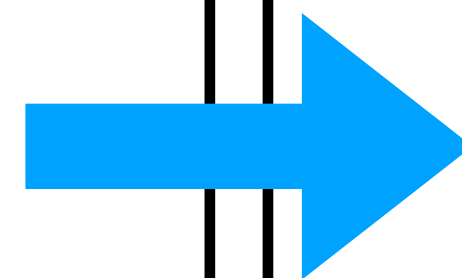
(実際にはファイルが数百個)



/data/www/www.pixiv.net

- index.php
- novel
 - show.php
- yyy.php
- zzz.php

(実際にはファイルが数百個)



www.pixiv.net/inc/

- IndexController.php
- Novel
 - ShowController.php
- YyyController.php
- ZzzController.php

手で移植するのは
ナンセンス

PHPカンファレンス2017

**技術****ビジネス****カルチャー・制度****イベント**[採用情報](#)

技術基盤チームの山際はWebサイトのURL設計と刷新の方針について話しました。

大規模Webサイトの URL刷新の方針と実装

2017/10/08 PHP Conference 2017

#エンジニア
#発表

作業時間は?

- ▶ クラス名の重複 => 数十個
- ▶ global関数の定義 => 400強

作業見積もり: 20時間超

作業時間は?

- ▶ クラス名の重複 => 数十個
- ▶ global関数の定義 => 400強

Vimで高速解決しました

クラス名のリネーム

```
syntax enable
let found = search('class')
if !found
    qall
endif
class_name = @0
" FooControllerをWww_View_FooControllerに置き換え
execute '%s/'.class_name.'/Www_View_'.class_name.'/ge'
```

クラス名をyank

クラス名を変更



@kana1

VimでPHPのコードをシュワルツ変換してソートする

PHP Vim リファクタリング

最終更新日 2017年12月12日 投稿日 2017年12月12日

(この記事は [ピクシブ株式会社 AdventCalendar 2017](#) の12日目の記事です)

今回のあらすじ



にゃんだーすわん

@tadsan · [フォローする](#)



リファクタリングDSLとしてのVim scriptの威力に再び社内がおののいてる

午後3:48 · 2017年10月25日



17



返信



リンクをコピー

[2件の返信を読む](#)

どうおののかせたかを紹介します。

問題

pixivのURLルート定義は以下のような形になっています:^[1]

```
lib/routes.php

function getUrlRouteMap()
{
    $route_map = [
        '/' => [
            'controller' => 'IndexController',
        ],
        '/discovery' => [
            'controller' => 'DiscoveryController',
        ],
        '/user/:user_id/series' => [
            'controller' => 'UserSeriesIndexController',
            'params' => [
                'user_id' => 'int',
            ],
        ],
        ...
        '/about.php' => [
            'controller' => 'AboutController',
        ],
        '/bookmark.php' => [
            'controller' => 'BookmarkController',
        ],
        ...
    ];
    return $route_map;
}
```

問題

pixivのURLルート定義は以下のような形になっています:^[1]

lib/routes.php

```
function getUrlRouteMap()
{
    $route_map = [
        '/' => [
            'controller' => 'IndexController',
        ],
        '/discovery' => [
            'controller' => 'DiscoveryController',
        ],
        '/user/:user_id/series' => [
            'controller' => 'UserSeriesIndexController',
            'params' => [
                'user_id' => 'int',
            ],
        ],
        ...
        '/about.php' => [
            'controller' => 'AboutController',
        ],
        '/bookmark.php' => [
            'controller' => 'BookmarkController',
        ],
        ...
    ];
    return $route_map;
}
```

見事に順番がバラバラです^[2]。これでは後からコードを読んだ人には何処に何が書かれているか分からない状態で良くありませんし、新しい定義を追加する時にも判断に迷うので各自が好き勝手な場所に書き始めてどんどんカオスになっていきます。数が少なければまだしも、このルート定義の数は800個以上あるという.....

というわけで **各ルート定義をURLの辞書順でソートする** というのが今回の目的です。

回答

```
./sort-routes lib/routes.php
```

sort-routes

```
# !/bin/bash  
vim -N -u NONE -i NONE -e -s -S "$0.vim" "$@"
```

sort-routes.vim

```
/\ $route_map = \[/  
mark a  
normal! %  
mark b  
  
'a+1,'b-1 g/'\./.*' => \[/./,/^ \{8}\],$/-1s/\n/XYZZY/  
'a+1,'b-1 sort  
'a+1,'b-1 s/XYZZY/\r/g  
  
update  
qall!
```

今も同じコードが
元気に走り続けてます



枯れた機能なので
積極的に弄る動機がない
(PHPのBC breakを踏まない限り)

Notes on Programming in C

[*Rob Pike*](#)

February 21, 1989

Introduction

Kernighan and Plauger's [*The Elements of Programming Style*](#) was an important and rightly influential book. But sometimes I feel its concise rules were taken as a cookbook approach to good style instead of the succinct expression of a philosophy they were meant to be. If the book claims that variable names should be chosen meaningfully, doesn't it then follow that variables whose names are small essays on their use are even better? Isn't `MaximumValueUntilOverflow` a better name than `maxval`? I don't think so.

What follows is a set of short essays that collectively encourage a philosophy of clarity in programming rather than giving hard rules. I don't expect you to agree with all of them, because they are opinion and opinions change with the times. But they've been accumulating in my head, if not on paper until now, for a long time, and are based on a lot of experience, so I hope they help you understand how to plan the details of a program. (I've yet to see a good essay on how to plan the whole thing, but then that's partly what this course is about.) If you find them idiosyncratic, fine; if you disagree with them, fine; but if they make you think about why you disagree, that's better. Under no circumstances should you program the way I say to because I say to; program the way you think expresses best what you're trying to accomplish in the program. And do so consistently and ruthlessly.

Your comments are welcome.

Issues of typography

A program is a sort of publication. It's meant to be read by the programmer, another programmer (perhaps yourself a few days, weeks or years later), and lastly a machine. The machine doesn't care how pretty the program is - if the program compiles, the machine's happy - but people do, and they should. Sometimes they care too much: pretty printers mechanically produce pretty output that accentuates irrelevant detail in the program, which is

Notes on Programming in C

ルール2: 計測すべし。計測するまでは速度のための調整をしてはならない。コードの一部が残りを圧倒しないのであれば、なおさらである。

— Rob Pike

<https://www.lysator.liu.se/c/pikestyle.html>

訳語は[UNIX哲学](#) - Wikipediaより引用
(2025年10月2日12:33:03版)

裏返せば、計測してみても
スループット悪化の主要因
と判断されない限り
安全性と開発者体験を優先

PHPには
型宣言の機能がある

処理系が実行時に 自動型チェックを実施

実行時？遅いのでは？



かつてはそうだった

言語のホットスポットは
処理系で改善されると
嬉しい

現代はJITが改善され
型宣言した方が
最適化の恩恵を受ける

当てずっぽうで
「速い」ものを選んで
問題解決にならない

これからの時代こそ
自我をもって
問題解決しよう