

ぶっちゃけどうなのPHP

Honestly, What's the Deal with PHP?



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
- 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)

今回のお題

PHPカンファレンス福岡2025

採択 2025/11/08 11:40～ Fusicホール レギュラートーク(30分)

ぶっちゃけどうなのPHP



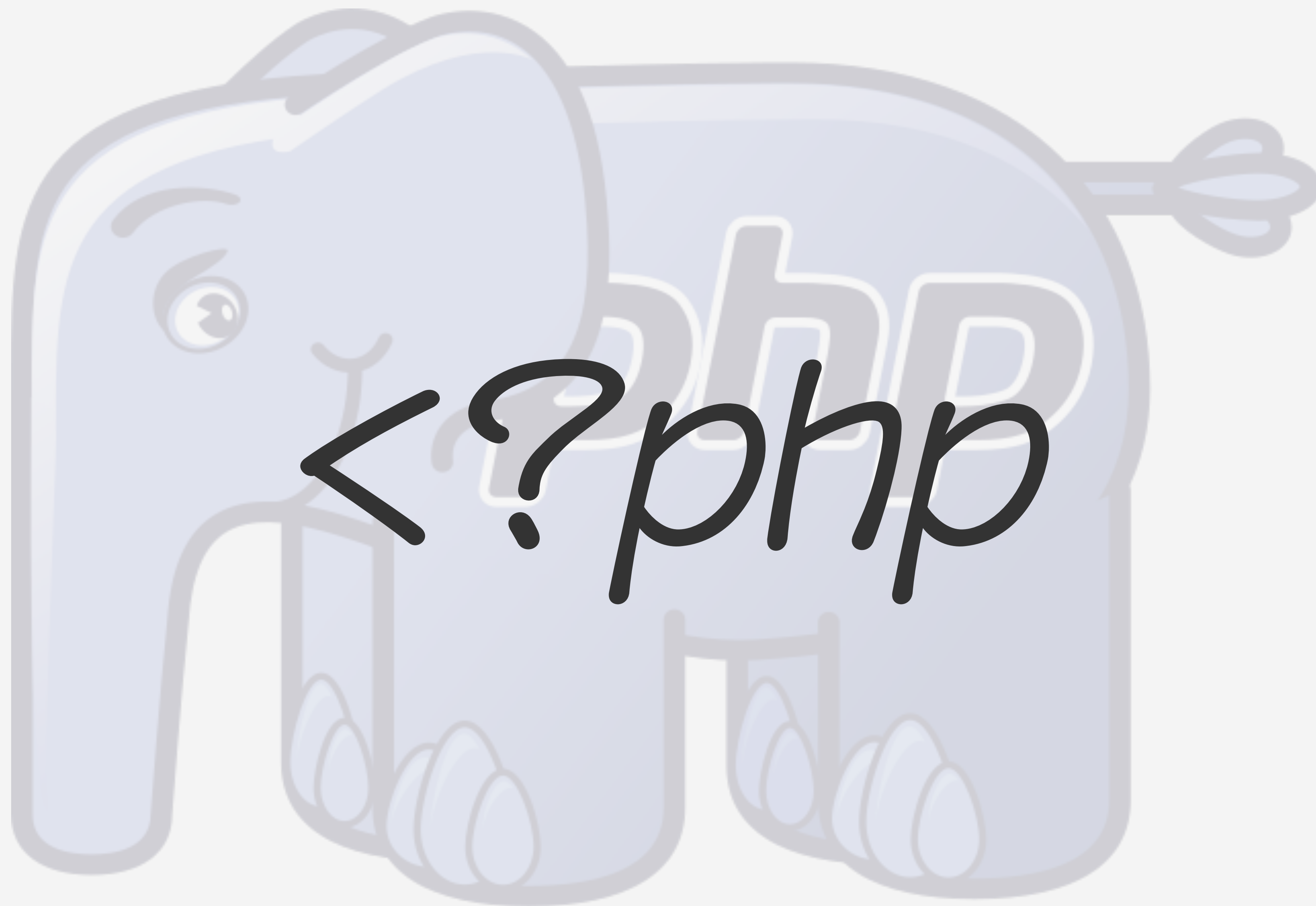
うさみけんた  tadsan

☆ 5

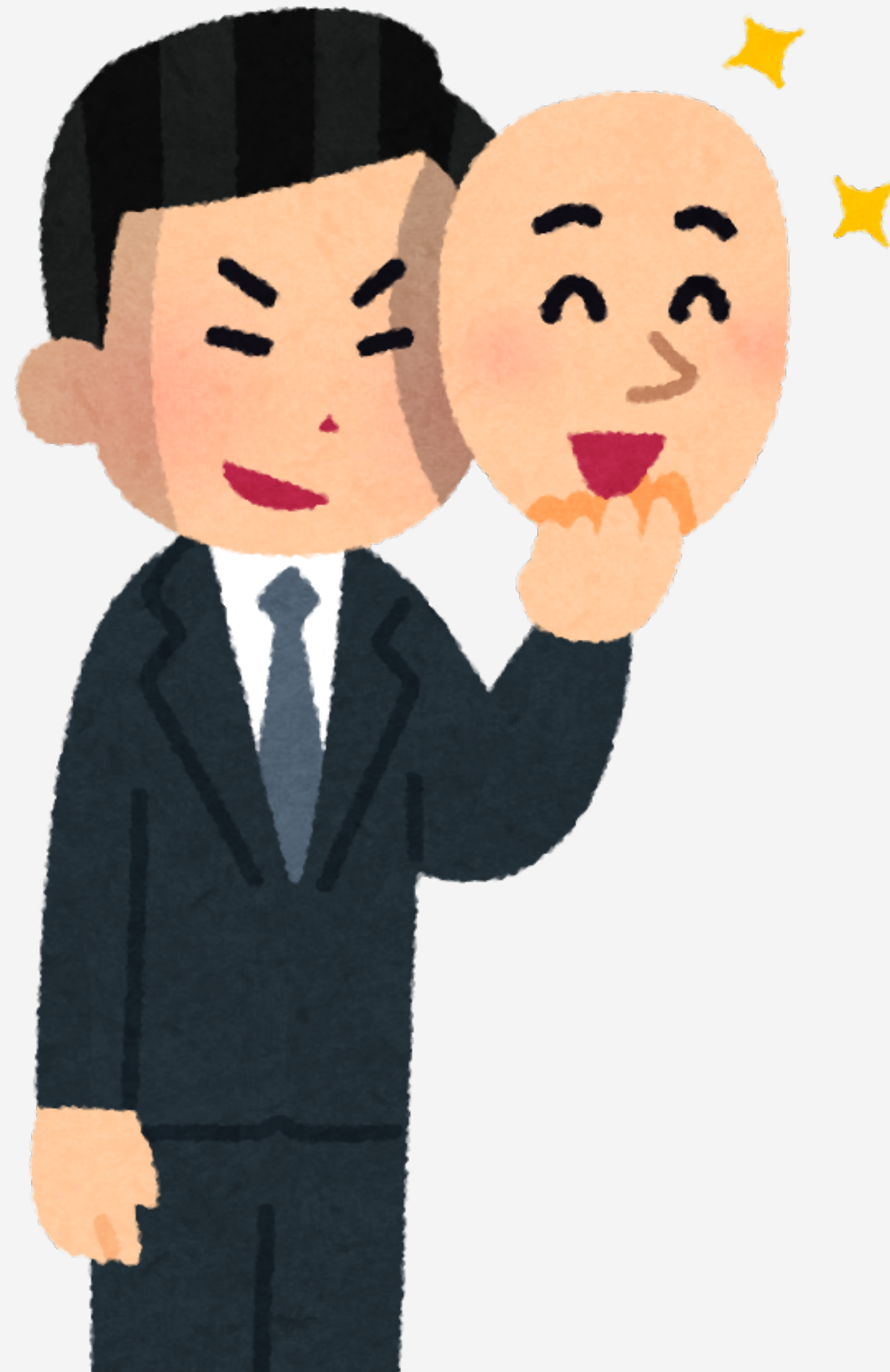
PHPが登場して30年、さまざまな言語が現れてはWeb開発に新たな可能性が開かれてきました。
新しい言語やフレームワークにはPHPが実現したものを取り込んだもの、野心的なパラダイムを打ち出したものも多くありますが、
しぶとくもPHPを完全に置き換えるには至っていません。

本トークでは歴史と多言語での実装事例を踏まえてPHPとWebの過去と現在の立ち位置を再確認して、未来の姿を占います。

- 前PHP時代 ～ Rasmusは何を革命したのか
- Webフレームワークの中世期断絶
- PHPが往かなかったWebフレームワークの世界線 ～ 普通の奴らの上を行け
- 非同期I/OとWeb ～ PHP最大の弱点と次世代への夢



PHP… ぶっちゃけどう思う？



おぼろげながら浮かんできたんです

Node.js... C#... ASP.NET...
Rust...
Go... Common Lisp...
Python... Java... TypeScript...
Ruby on Rails...
Kotlin... Swift... Smalltalk...

いろいろな選択肢が
あるのに我々は
PHPにしがみつくなのか

個人的には楽しい
言語をいろいろ
選んできた

俺 vs プログラミング言語



- 2005-2009: PHS(携帯電話)でぽちぽちHTMLとJSを書いていた
- 2010: 札幌でRubyコミュニティに出入りしてた
 - この頃からプログラミング言語論とか言語処理系に興味を持つ
- 2011: 札幌でPython勉強会に参加したり、F#を教わる
- 2012: 東京に引っ越してRubyKaigi, Shibuya.rbなどに出入り
- 2016-: PHP系のカンファレンス発表、Emacsのイベント運営

仕事では12年くらい
登壇は9年くらい

プログラミング言語と Webとフレームワーク にこだわりのある人間

アジェンダ

- 前PHP時代 ~ Rasmusは何を革命したのか
- Webフレームワークの中世期断絶
- PHPが往かなかったWebフレームワークの世界線
- 非同期I/OとWeb

話は
1993年に
さかのぼる

出典: フリー百科事典『ウィキペディア (Wikipedia)』

ラスマス・ラードフ（Rasmus Lerdorf, 1968年11月22日 - ）は**デンマーク系カナダ人**の**プログラマ**で、**プログラミング言語PHP**の最初のバージョンであるPHP/FIの開発者。また、**アンディ・ガトマンズ**、**ゼーブ・スラスキー**によるPHPの後継バージョンの開発にも携わり、その発展に寄与した。

出生地はグリーンランド。1993年にカナダの**ウォータールー大学**においてシステムデザイン工学分野での応用科学学士号 (BASc) を取得し、同大学を卒業。2002年9月より2009年11月6日まで**Yahoo!**の**技術者**として働く。

外部リンク [[ソースを編集](#)]

- lerdorf.com - 個人サイト



ラスマス・ラードフ



ウィキクォートに**ラスマス・ラードフ**に関する引用句集があります。

[表](#)・[話](#)・[編](#)・[歴](#)

PHP

[\[表示\]](#)

典拠管理データベース

[\[表示\]](#)

カテゴリ: [カナダのプログラマ](#) | [1968年生](#) | [存命人物](#) | [ケケタリク出身の人物](#)

出典: フリー百科事典『ウィキペディア (Wikipedia)』より引用
<https://ja.wikipedia.org/wiki/ラスマス・ラードフ> 2021-07-01T14:10:55版

[はてな匿名ダイアリー](#) > [anond:20100427231539](#)

< <http://anond.hatelabo.jp/20100...> | [フィクションに興味がある](#) >

2010-04-27

■ 伝説のPHP作者「Rasmus Lerdorf」名言集を聞くと嫌PHP厨がファビョる

- 今のPHPを作ったのは、何十人もの開発者ですよ。私は1人目の開発者だったに過ぎません。
- 問題を解くのが好きなだけで、プログラミングは大嫌いです。
- いかにプログラミングを避けるかを考えていたら、コードを再利用するためのツールとしてPHPができました。
- PHPは、歯ブラシみたいなものですね。毎日使うものですけど、だから何でしょう？ 誰が歯ブラシの本なんて読みたがります？
- パーザを書くのは苦手です。本当にダメなんです。今でもね。

2010年04月17日 20:40 📁 その他

伝説の喫煙者「黒木 灰次郎」名言集を聞くと嫌煙がファビョる 1 user

引用元：<http://yutori7.2ch.net/test/read.cgi/news4vip/1271431628/>

1 名前： 以下、名無しにかわりましてVIPがお送りします 投稿日： 2010/04/17(土) 00:27:08.66 ID:jCpGzqAR0

タバコを吸うと料理の味がわからなくなる？

タバコを吸わない人にタバコの味はわかりませんよ。

2 名前： 以下、名無しにかわりましてVIPがお送りします 投稿日： 2010/04/17(土) 00:28:35.40 ID:jCpGzqAR0

僕は空気よりタバコを吸ってる時間の方が長い

5 名前： 以下、名無しにかわりましてVIPがお

一日に何本吸ってるか？

何キ口吸ってるかならわかりますけど

32 名前： 以下、名無しにかわりましてVIPがお送りします 投稿日：

あれ、これってバスタブなんですか？灰皿だと思ってました。

37 名前： 以下、名無しにかわりましてVIPがお送りします 投稿日：

喫煙者と言えないですよ。肺しか黒くなってないうちは。

38 名前： 以下、名無しにかわりましてVIPがお送りします 投稿日：

死んだら地獄に行きたいですね。天国ってきっと禁煙でしょ？

出典:伝説の喫煙者「黒木 灰次郎」名言集を聞くと嫌煙がファビョる
<http://blog.livedoor.jp/hokanisurukotoha/archives/1121714.html>

≡ ラスマス・ラードフ

🌐 3の言語版 ▼

本文 [トーク](#)

[閲覧](#) [編集](#) [履歴表示](#) ツールボックス ▼

ラスマス・ラードフ（Rasmus Lerdorf, 1968年11月22日 - ）はデンマーク系カナダ人のプログラマで、プログラミング言語PHPの最初のバージョンであるPHP/FIの開発者。

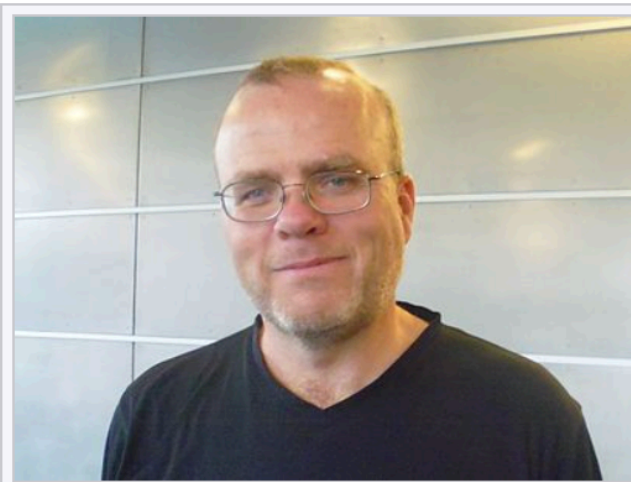


ウィキペディアにも**ラスマス・ラードフ**の記事があります。

出典が明らかなもの [\[編集\]](#)

[はてな匿名ダイアリー](#)に記載された英語版の翻訳 [🔗](#)より移植

- 今のPHPを作ったのは、何十人もの開発者ですよ。私は1人目の開発者だったに過ぎません。
 - [sitepoint.com](#) [🔗](#)
- 問題を解くのが好きなだけで、プログラミングは大嫌いです。
 - [sitepoint.com](#) [🔗](#)
- いかにプログラミングを避けるかを考えていたら、コードを再利用するためのツールとしてPHPができました。
 - [Itconversations.com](#) [🔗](#)
- PHPは、歯ブラシみたいなものですね。毎日使うものですけど、だから何でしょう？誰が歯ブラシの本なんて読みたがります？
 - [sitepoint.com](#) [🔗](#)
- パーザを書くのは苦手です。本当にダメなんです。今でもね。
 - [Itconversations.com](#) [🔗](#)
- PHPには「protected属性」も「仮想メソッド」もありますよ。情報学科の教官が「重要だ」っていうやつは何でもね。僕自身は、こんなものどうでもいいと思ってますけど。
 - [Itconversations.com](#) [🔗](#)
- プログラミングを好む人があるのは知ってますが、全く理解できないですね。
 - [Itconversations.com](#) [🔗](#)
- 僕はホンモノのプログラマではありませんから、やっつけ仕事ですよ。ホンモノのプログラマは、「動いてるように見えるけど、メモリリークだらけじゃないか。直す必要があるかもね」なんて言うでしょう？僕なら、10リクエストごとにApacheを再起動しますね。
 - [Itconversations.com](#) [🔗](#)



ラスマス・ラードフ



出典: ウィキクォート <https://ja.wikiquote.org/wiki/ラスマス・ラードフ>

PHPは多くの人から疎まれ、
まじめなプログラミング言語として
考察対象とされていない節がある

Rasmus Lerdorf

Yahoo

PHP on Hormones

MySQL Conference

44 minutes, 20.5mb, recorded 2007-04-26

Topics: [Software Development](#)



[Download Audio](#)

In 1993, when Rasmus first saw the Mosaic Web browser, he knew that the Internet would be the platform of choice. But his employer, a Brazilian company, did not pay heed so he quit to return to Canada to do consulting work. During this six-month period, he found himself repetitively writing the same CGI programs in C. To avoid repetition, he collected his library of C programs and added a template parser that parsed HTML and made calls to his C routines. Thus was born the first version of PHP.



Rasmus believes there are four kinds of programmers. First, the pragmatic ones who are just after solving their own problems. The second kind finds programming as a means of self-expression, like an artist finds self-expression in his art-work. The third are the real programmers who enjoy programming for its own sake because it creates a hormone called oxytocin in them, and the fourth are the open source zealots who wish to change the world. He claims to be of the first kind. He programs to solve his problem and then moves on. He confesses that he created PHP purely to serve his own interest, to solve his own set of problems. He made the source publicly available so others could benefit from it. That set the ball rolling. Today, PHP runs a considerable number of some of the largest websites on the planet.

Hear the story of the evolution of PHP from being a purely procedural language to its current state of a full-fledged object oriented language, from its creator Rasmus Lerdorf. In this presentation, Rasmus also talks about the performance of PHP, profiling, security issues and vulnerabilities that websites are prone to, how to tackle them to some extent, and about his love for API. The [slides for this session](#) contain code snippets of the old and new PHP versions and also of the Flickr and Yahoo! Maps API examples.

Rasmus Lerdorf is the original creator of the PHP programming language, the mod_info Apache module, and the ANSI92 SQL-defying LIMIT clause in mSQL 1.x which has now crept into both MySQL and PostgreSQL.

Prior to joining Yahoo! as an infrastructure engineer in 2002, he was at a string of companies including Linuxcare, IBM, and Bell Canada working on internet technologies.

Resources

- Slides for the session
- Rasmus Lerdorf's website

Rasmus LerdorfはMySQLとPostgreSQLの両方に導入されているSQL標準(ANSI92 SQL)を無視したmSQLのLIMIT句のオリジナル作者です

This free podcast is from our MySQL Conference series.

皆さん、おはようございます。こんな時間に起きてくださってありがとうございます。Rasmus は PHP の父です。これはすごいことです。私たちが話しているのは、地球上でかなりの数の大規模な Web サイトを運営している PHP です。宣伝活動が盛んであるにもかかわらず、他の言語にも予算やマーケティング担当者などが集まっています。

PHP は言語として独自の地位を確立しており、オープンソースで見られる最もダイナミックかつクリエイティブな要素の 1 つであると考えられます。それでは、これ以上は省略して、Rasmus を紹介しましょう。

[拍手]



PHP on Hormones

MySQL Conference

Apr.26, 2007. Santa Clara

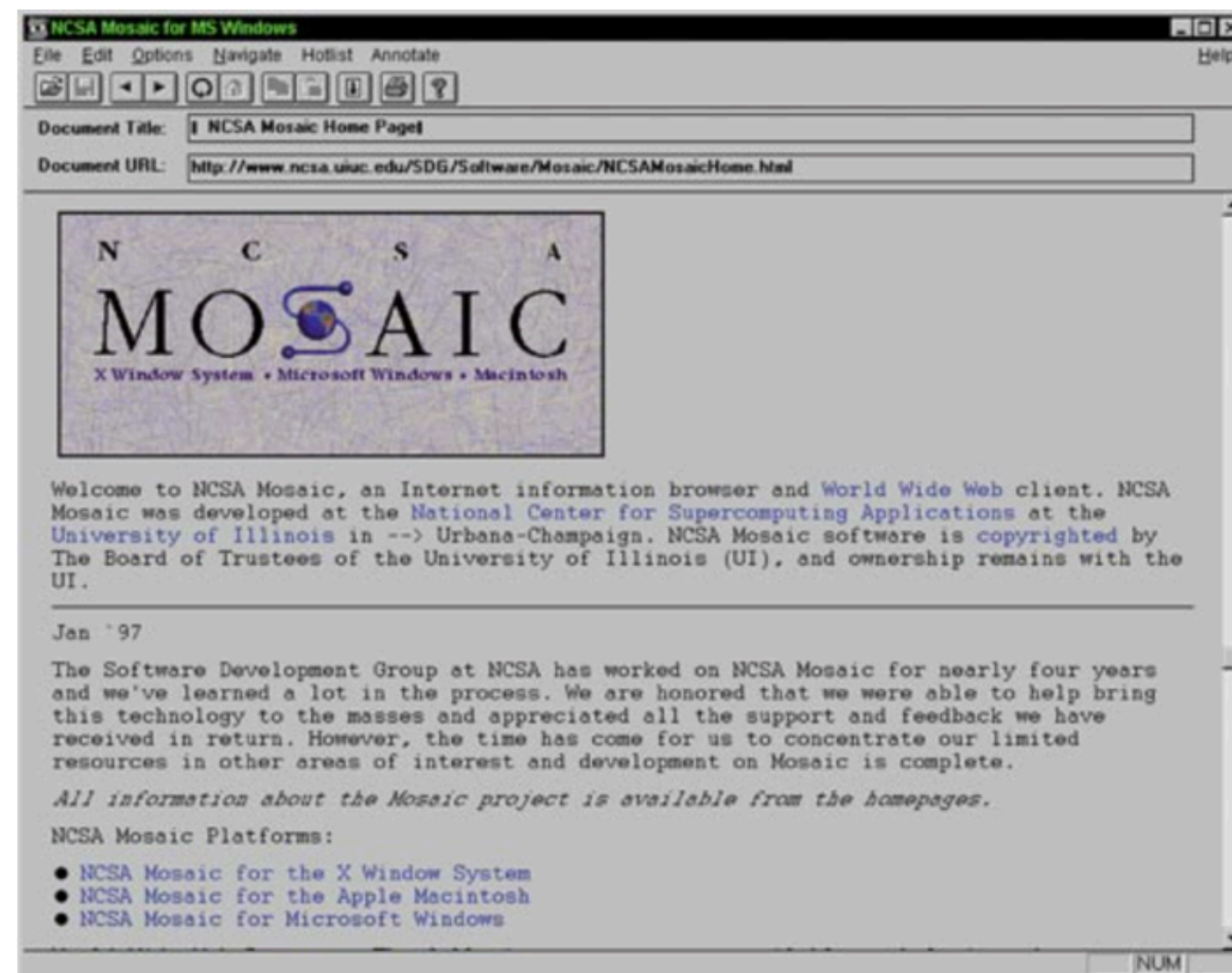
Rasmus Lerdorf <rasmus@php.net>

<http://talks.php.net/show/mysql07key>

そして、初めて Mosaic Browser を見たとき、私はブラジルの会社のオーナーに電話をかけて、「みんな、今すぐ会社全体を変えて、インターネット以外のことはやらないようにしなきゃいけない。このブラウザは世界を席巻するだろうから」と言いました。彼らは同意しなかったので、私は辞めました。



1993



私はカナダに戻り、ウェブコンサルティングを始めました。当時のウェブページはこんな感じでした。



1993

```
<HTML>
  <HEAD>
    <TITLE>My Personal Home Page</TITLE>
  </HEAD>
  <BODY>
    This is my cool page<P>
    And look at my counter<P>
    <IMG SRC="/cgi-bin/counter.pl">
  </BODY>
</HTML>
```


それらの背後にあるコード。背後にあるわけではなく、こんな感じです。ホームページがあり、誰もがページの下部にくだらないカウンターを持っていました。

彼らは、「これだけの訪問数、これはちょっとしたグラフィックです」と言いました。私は通常 Perl で書いていたので、このような小さなcgi-binカウンターでした。本当に、あまり印象的ではありませんでした。そして、それらのカウンターは、それをそこに置いた人以外には意味がありませんでした。

当時、私が雇われたのは、動的なウェブページを書いたり、バックエンドのデータベースや、企業や大学がかつてオンラインに公開していたあらゆる情報を取得したりするためでした。彼らは、その構築を手伝うために私を雇ったのです。そして、私は同じコードを何度も何度も書くことになりました。

私は C で多くの CGI プログラムを書きました。そしてしばらくして、同じ C コードを何度も書き続けることに本当に疲れてしまいました。

そこで、過去 6 か月ほどの間に書いた C コードをすべて収集し、ライブラリにまとめました。そして、HTML テンプレートを解析して C コードを呼び出す、非常にシンプルなテンプレート パーサーを考案しました。

CGIとは何か

- **Common Gateway Interface**の略、1993年頃に登場
 - HTTPサーバを独自実装しなくても、シンプルなスクリプトを書くだけで動的ページを実現できる
 - ユーザーからリクエストが来る度にプロセスを起動する
 - シェアードナッシング（並行して処理するリクエストと干渉しない）

CGIとプログラミング言語

- 仕様上、標準入出力(stdio)と環境変数を使える言語ならなんでもいい
- 開発の利便性などから、Perlスクリプトが普及した
 - 文字列処理のしやすさ、正規表現、cgi-lib.pl、言語としての書き心地…
 - C言語での文字列処理は本質的に難しい
- ラスマスが重要視したのはそこではなかった
 - パフォーマンス、HTMLとSQLとの統合…

これは基本的に 1994 年頃の PHP バージョン 1 です。



1994

```
<!--getenv HTTP_USER_AGENT-->
<!--ifsubstr $exec_result Mozilla-->
Hey, you are using Netscape!<p>
<!--endif-->

<!--sql database select * from table where user='$username'-->

<!--ifless $numentries 1-->
Sorry, that record does not exist<p>
<!--endif exit-->

Welcome <!--$user-->!<p>
You have <!--$index:0--> credits left in your account.<p>

<!--include /text/footer.html-->
```


Rasmusは何をしたか（～1995年くらい）

- パフォーマンスのため、C言語でCGIを書いた
- 自分で書いたCGIツールに「PHP」という名前をつけた
- CGIをやめ、Apache HTTP Serverのモジュール化した
- ソースコードごとフリーソフトウェアとして公開した
 - Perl, Apacheなどの先例はあるが、「オープンソース」ブーム前

今のところ、PHP の見た目は現在のものとあまり似ていません。しかし、見れば何をしているのかは大体わかります。

たとえば `getenv`、ユーザー エージェントを取得するには、Mozilla かどうかを確認します。すると、彼は「ねえ、あなたは Netscape を使っていますね」と言いました。

そこには SQL クエリがあります。最初のバージョンにも SQL クエリがありました。当時、私はオーストラリアの David Hughes という人が作成した Mini SQL というデータベースを使用していました。

SQL クエリも少し使用しました。ロジックも少し使用しました。少ない場合は、中括弧がないことがわかります。

セミコロンはありません。非常にシンプルなテンプレート言語です。

当時は、基本的にこれを使用していたのは私だけでした。気軽に変更を加えることができました。毎日、言語を大幅に変更していました。そして、それほど時間はかかりませんでした。

1993年時点で存在したデータベースソフトウェア。
フリーソフトウェアではないがソースコードが
公開されていて、MySQLなどに影響を及ぼした。

1995 年、1995 年中頃までに、PHP は次のようになりました。これは、実際のところ、今日の PHP とそれほど変わりません。



1995

```
<?
$name = "bob";
$db = "db";
$result = mysql($db,"select * from table where firstname='$name'");
$num = mysql_numrows($result);
echo "$num records found!<p>";
$i=0;
while($i<$num);
    echo mysql_result($result,$i,"fullname");
    echo "<br>";
    echo mysql_result($result,$i,"address");
    echo "<br>";
    $i++;
endwhile;
>
```


ここではいくつかの違いがあることが簡単にわかります。中括弧がないことに気がつくでしょう。

この `while` ループは、後ろにセミコロンが付いているので、構文エラーのように見えます。しかし、私はパーサーを書くのが本当に苦手でした。今でもパーサーを書くのは本当に苦手です。

`while` つまり、`and while . if and`、などと書かれたトークンがあると、読みやすくなります `if`。つまり、どの構造を閉じているのかが実際にわかりました。つまり、中括弧はまったく必要ありませんでした。

2005

```
<?php
class db {
    protected static $dbh = false;

    function connect() {
        self::$dbh = new PDO('mysql:host=localhost;dbname=test','user','pass');
        self::$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }
}

class items extends db {
    function load($name) {
        if(!self::$dbh) $this->connect();
        try {
            if(!self::$dbh) $this->connect();
            $stmt = self::$dbh->prepare("SELECT * FROM items WHERE firstname=:name
                                      ORDER by ctime desc");

            $ret = $stmt->execute(array('name'=>$name));
        } catch (PDOException $e) {
            die($e->getMessage());
        }
    }
}
```

今から 10 年後、いや、それより 12 年後だと思います。しかし、2005 年の時点では、同じコードがこのような状態になっています。これが 10 年間の開発に値するかどうかは、私にはよくわかりません。

しかし、コンピュータサイエンスの人々は、これですべてがオブジェクト指向になったことに満足しています。これはクールだからです。保護されたプロパティのようなものがあります。抽象メソッドがあります。コンピュータサイエンスの先生が本当に使用すべきだと言ったものがすべてあります。実際のところ、私はこのようなくだらないことにはまったく関心がありません。

私は問題を解決するツールを作ることに関心があります。プログラミングは大嫌いです。プログラミングを避けるために PHP を書きました。できるだけ早く問題を解決したいのです。そして、私が PHP を書いた方法は、私が知っていることに基づいていることに気づきました。そして、他の人がすでに知っているだろうと思っていたことに基づいており、学習曲線が非常に暗くなっています。

今日の大学を出た若者は、これが彼らの知識です。彼らはオブジェクト指向プログラミングを知っています。これが彼らにとって馴染み深いものです。手続き型プログラミングとは何かを説明しなければならない人たちがいました。

今日の大学を出た若者は、これが彼らの知識です。彼らはオブジェクト指向プログラミングを知っています。これが彼らにとって馴染み深いものです。手続き型プログラミングとは何かを説明しなければならない人たちがいました。

私は彼らに関数呼び出しを見せました。それで、クラスはどこにあるのですか？これはどこから来るのですか？つまり、私はこの手続き型呼び出しを理解していません。

それはただそこにあり、理解できないことをしているだけです。私にとっては、それはちょっと目を見張るものでした。そして、PHP が少し変化し、現代のオブジェクト指向言語に似たものへと進化したのは、人々がそのようなプログラミング方法を期待しているからです。

これが、私たちが PHP の世界で行っていることです。もちろん、PHP はオープンソースです。

私は長年、オープンソースの仕事をしてきました。いつも聞かれるのは、なぜコードを公開するのか、なぜこのようなことをするのか、ということです。

そして、世界には 4 つの異なるタイプのオープンソース開発者がいると私は考えています。

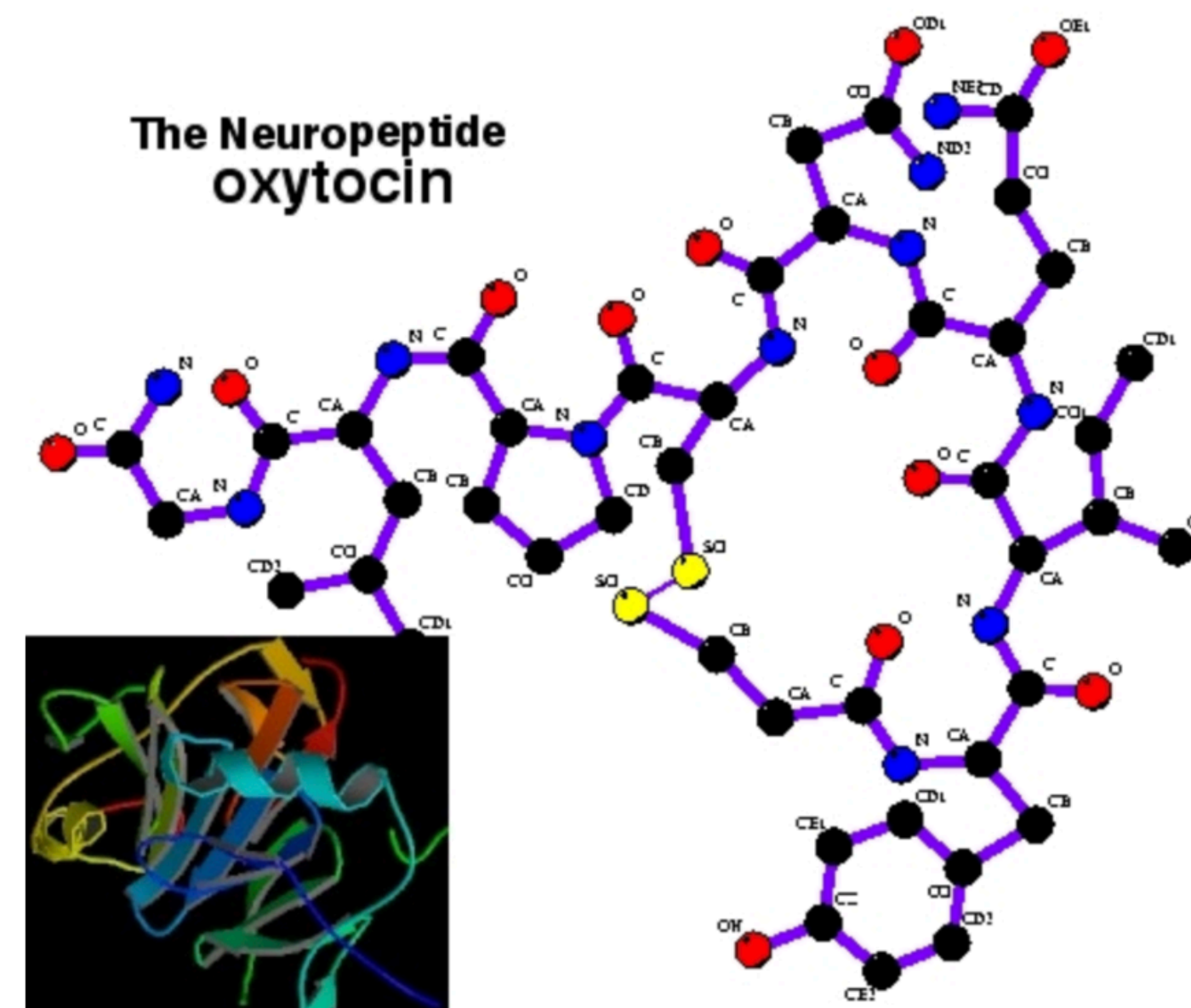


Open Source



Why do people contribute?

- Self-interest
- Self-expression
- Hormones
- Improve the World



Nature's Trust Hormone

私はまさにこのスライドの最初のタイプ、つまり純粋に自己利益を追求するタイプです。問題を解決するツールが必要でした。私の興味はツールにはあまりありません。私の興味は問題を解決することにあります。

当時私を雇ってくれたさまざまな人たち、中でも大きなトロント大学は、Web 駆動型の大規模なダイヤルアップ システムの構築を私に依頼しました。彼らは私がどのようにそれを実現するかは気にしませんでした。彼らが求めていたのは、ソリューションだけでした。

ですから、途中でツールをオープンソース化したいと思ったとしても、それはまったく問題ありません。PHP の多くは、実際にはトロント大学向けに書かれたものです。ですから、私にとっては、それは純粋に自己利益のためでした。

オープンソースにしたのは、他の人たちが私に「どうやってそれをやったの？」と聞いてきたからです。私は「私はこうやってやった。これが私のツールだ」と答えました。彼らはそれを採用し、私にパッチを送り始めました。

トロント大学は、私が驚くべき速さでコードを書いていたことを素晴らしいと考えました。寝ている間もバグを修正していました。毎朝メールボックスを見ると、世界中から、自分が抱えていることすら知らなかったバグの修正が 3 ～ 4 件届いている、と気づくのです。素晴らしいシステムだと思いました。

同じ問題を抱える世界中の人たちと集まってみたらどうでしょう。そして、私たち全員が集まって、この共通の問題を解決するツールに取り組みます。そして、私たち一人ひとりが、自分の小さな世界の中では優秀に見えました。なぜなら、私たちはこのツールを非常に速く構築し、世界中の人々がそれに取り組んでいるという規模を利用していたからです。

ですから、これはソフトウェア開発、つまり私が必要としていたツールの開発にとって本当に良いモデルだと思いました。

他の人には別の考えがあります。つまり、他の人は自己表現のために貢献しているのです。実際にプログラミングが好きな人もいます。なぜ彼らがプログラミングが好きなのかは理解できません。

しかし、プログラミングが好きな人は、画家が絵を描いて他の人に見せるのと同じように、自分を表現したいと思っています。ミュージシャンは、それが自分を表現する方法なので、人前で演奏したいと思っています。プログラマーは、自分のコードを他の人に見せたいと思っています。

- 「やあ、私は優秀なプログラマーだ。」
- 「このコードを見てください。」
- 「この問題を私が解決したこの複雑な方法を見てください。」
- 「私が考え出したこのクールなアルゴリズムを見てください。」

そして、それが彼らの自己表現の仕方なのです。そして、私はそれが素晴らしいことだと思います。なぜなら、本物のプログラマーがいなければ、PHP は今日存在していなかったでしょうから。

そして、それが彼らの自己表現の仕方なのです。そして、私はそれが素晴らしいことだと思います。なぜなら、本物のプログラマーがいなければ、PHP は今日存在していなかったでしょうから。

私は本物のプログラマーではありません。動作するまでいろいろと試して、それから先に進みます。

本物のプログラマーはそこに座って「はい、動作します」と言うでしょう。しかし、メモリがあらゆる場所でリークしています。おそらく、これを修正する必要があります。

10 リクエストごとに Apache を再起動するだけにしようと思います。リークを修正しましょう。私は細かいことにこだわるタイプではありません。つまり、動作させるだけです。いくつかアイデアを思いつきます。

他の人たちがこれを本当に役に立つものにしてくれます。

ホルモン、これはオキシトシンというとても素晴らしいホルモンです。基本的には自然の信頼ホルモンです。そして、他の人とあまりうまく交流できない子供やオタクがたくさんいます。しかし、これらの子供は他の人と交流する必要があります。そして、他の人と交流すると、オキシトシンというホルモンが分泌され、気分が良くなります。これは、男性と女性の両方で、出産時やオーガズム時に分泌されるのと同じホルモンです。そして、他の人と交流すると、このホルモンも分泌されます。

たとえば、オタクたちが World of Warcraft をプレイしているのを目にするでしょう。彼らをオタクと呼ぶつもりはありませんが、内向的な子供たちが World of Warcraft をプレイしているのをよく見かけます。これが特徴の1つです。これらの大規模でインタラクティブなオンラインゲームでは、子供たちは実際に他の人と交流します。そして、それが彼らのオキシトシンを分泌させます。

そして、ゲームにそれほど興味がない他の子供たちは、実際にオープンソースプロジェクトに参加して貢献し、そのようにして他の人と話します。それが彼らの仲間との交流の方法です。

彼らはバーに行ったり、友達と野球をしたりしていると思います。コンピューターの前に座ってオープンソースの仕事をしています。そして、世界を良くするためだけに何かをする変人もいます。

たとえば、私は世界をより良くするためにオープンソースに取り組むつもりです。オープンソースの多くは世界をより良くすると思います。これは素晴らしい副次効果です。しかし、PHP で世界をより良くしようと決めたと言ったら、それは本当に嘘になります。

いいえ。ツールが必要でした。他の人はそれが便利だと思いました。素晴らしい。それで、これはあなたが気にする何かとどのように関係しているのでしょうか？

オープンソース プロジェクトを運営する上で、この言葉はまさに真実であると思います。すべてを自分のことだけに考えていては、まともなオープンソース プロジェクトを運営することはできません。オープンソース プロジェクトに貢献する人々を中心に据える必要があります。

このプロジェクトに関わっている人たちが自分自身についてどう感じているかを考えなければなりません。ある程度の所有権を放棄しなければなりません。これは彼らのプロジェクトであり、みんなで一緒に取り組んでいるのだという意識を彼らに与えなければなりません。そして、それは私にとって当初最も困難なことの1つでした。なぜなら、それは私の仕事だったからです。

PHP は私の宝物でした。コントロールを放棄し、基本的に一步下がって、誰に対しても特別な権限を持つことなく、他のみんなと同じように PHP に貢献していました。

もし誰かが私に反対したら、私たち3人と2人が「いやいや、私たちはこうしなければなりません」と言ったら、私は完全に反対しました。

それは問題ではありません。私が始めたからといって、私がそれに関して大きな発言権を持つわけではありません。そしてそれは最初よりも少し大変でした。しかし、私は本当に怠け者です。

そして、もし他の人々がそのことについて本当に強い感情を抱いているなら、私は彼らと戦うのが面倒くさすぎるのです。皆さんが少しは気にしているかもしれない世界でも同じことが言えると思います。あちこちにあるこれらのサイトはすべて、同じ特徴を示していると思います。

Q. つまりPHPって
何だったの？

A. 異常に行動力があるRasmus
さんが自分のためにPHPを作っ
て、何回か書き直した後に利己的
な動機でソースコード公開したら
世界的に開発が盛り上がった



pixiv

[Publicationへの投稿](#) Edit

PHP Early Days (誕生から4まで)

2023/09/17に公開

 PHP

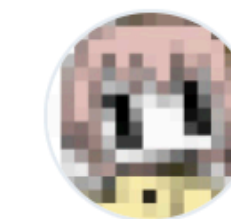
php4

 Tech

この記事は何か

某社の某雑誌に寄稿しようと思って書いた記事の初稿... だったのですが、テーマじゃないところに紙幅を割きすぎて本来のテーマを大幅に逸脱した上に、書いてから三年ほど置いてしまったものです (執筆自体はまだ諦めてません)。

ちょうど「LLイベント」こと[Learn Languages 2023](#)というイベントで「**PHPの20年とこれから**」としてPHPの歴史を話したので、せっかくなのでこの期に公開しましょう。



にゃんだーすわん



にゃーん

[バッジを贈る](#)[バッジを贈るとは →](#)

目次

[● この記事は何か](#)[Webという話について](#)

PHPはWebに何をもたらしたか

- HTMLテンプレート（HTML内に制御構造や変数展開）
 - JSP(Java), eRuby/ERBなどのHTMLテンプレートに影響
 - それ以前は `print "<div>Hello</div>";` のようなコード
- CGIの代替としてのサーバーモジュール組み込みと開発者体験
 - 現代ではコンパイルが必要な言語でも「ホットリロード」が普及
- フリーソフトウェアとして公開された言語(の先行事例のひとつ)

第2部

Webフレームワークの 中世期断絶

あるいは
幼年期の終わり

コードを書くとは
何らかの単位で
抽象化したくなる

コードが増えると
バグも増える

コードが減れば
バグも減る

ライブラリ化
フレームワーク化
レイヤー化

ライブラリ化

- 処理を関数やクラスの形でまとめて呼び出せるようにする
- ある人曰く「PHPの関数の数は、人間がWebでやりたいことの数」である
 - 完全に同意するわけではないが、本質の一面を捉えていると思う
- PHPの関数は外付けのライブラリに留まらず、HTTPをネイティブサポート
 - これは良し悪しあるが、セキュリティに関しては単なるCGIライブラリでは実現できないレベルでヘッダイнジェクションの対策ができる

フレームワーク化

- ライブラリとの最も明確な違いは「制御の反転」
 - あなたはコードを書いてライブラリ関数を呼び出す
 - フレームワークはあなたの書いたコードを呼び出す
- どのような規則でコードを実行するかはフレームワークの性質による
 - Webフレームワークはユーザーのリクエストによって対象を起動
 - テストフレームワークはディレクトリやファイル名によって対象を起動

レイヤー化

- 処理を「層(レイヤー)」に分けて整理する
- これにはさまざまな観点がある:
 - MVC (Model-View-Controller)
 - 3層アーキテクチャ、Ports and Adapters(クリーンアーキテクチャ)
 - HTTPインフラとロジックの分離

HTTP層のレイヤー化

- PHPはHTTP機能を言語が直接サポートする(Server API)
 - mod_php, php-fpm, cgi, cliなどがあるが、あまり意識の必要がない
- 他言語では従来のCGIスクリプトは廃れて、サーバとアプリケーションを分離するインターフェイスを定める方向にシフトした
 - Python=WSGI, Ruby=Rack, Perl=PSGI
- GoやNode.jsも標準でHTTPサーバをバンドルしている

Ruby on Railsの登場

- Webフレームワーク界の風雲児・フルスタックフレームワーク
 - CoC（設定より規約）：重厚な設定が必要なFWのアンチテーゼ
設定を書かなくても慣習通りに配置すれば勝手に呼び出してくれる
 - 多数の高機能な組み込みライブラリ：
O/Rマッパー、ルーター/URLヘルパー、
 - 積極的なコード生成(スキヤフォルディング)：
最初期は「15分でブログを作れる」と宣伝された（2005年）

RubyにとってRailsは何を変えたか

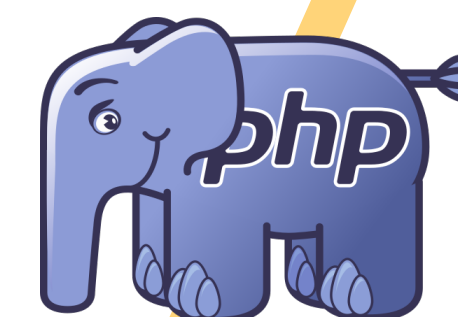
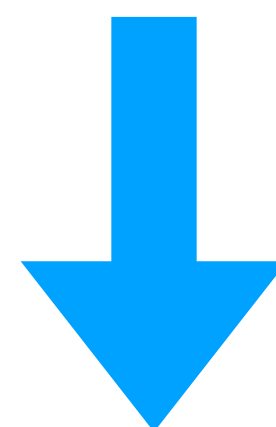
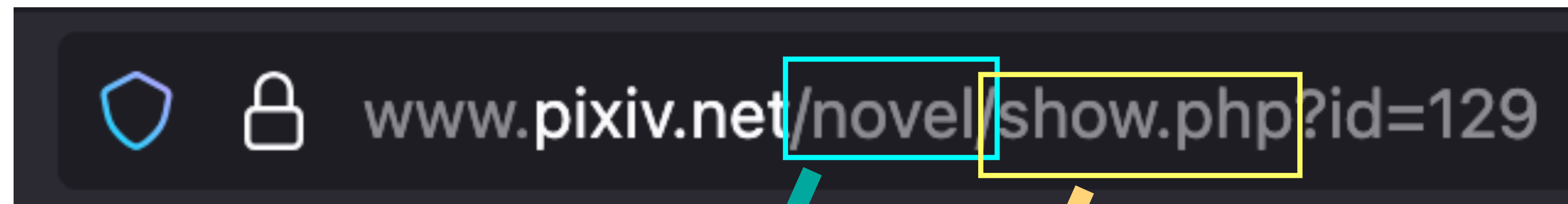
- Rails 1.0は独自のHTTPレイヤーで実装されていた
- 現代のRails(Rack)アプリケーションをCGIで実行することは可能だが、オーバーヘッドが大きい
- CGI時代のRubyスクリプトと、RailsのControllerクラスの実装は互換性がない
- 「コード書き直し」 or 「捨てる」の二択を迫られる

PerlとHTTPレイヤー

- 2000年代まで「WebアプリといえばPerlでCGI」という感じだった
 - SNS以前はKENT WEBの掲示板やチャットをレンタルサーバーに置いてコミュニケーションをとっていた時代もあった
- CGI時代のPerlスクリプトと、業務で開発するPerlアプリケーションの世界は分たれてしまった…

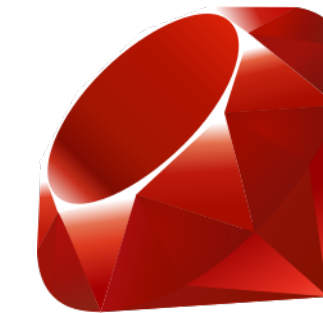
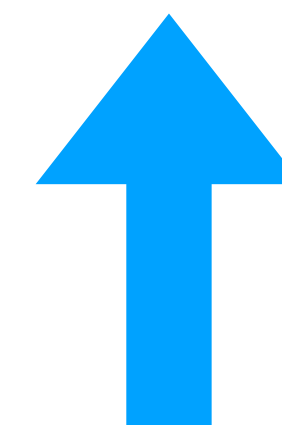
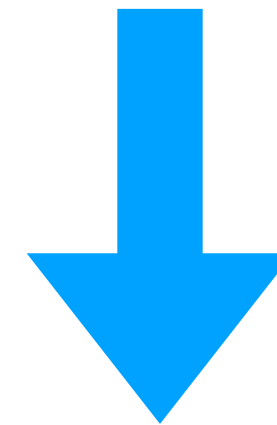
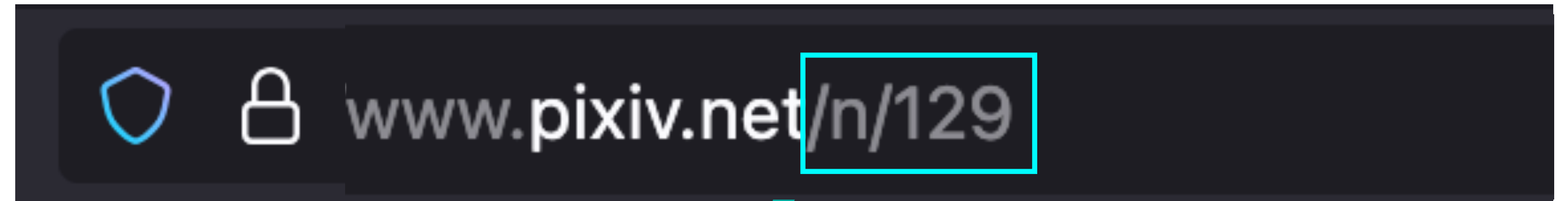
PHP界への余波

- 2004年を画期として、「フレームワーク」というものの見方が変わった
- CakePHPは当時の機能で再現可能な範囲で最大限Railsを模倣している
 - ルーティングすら書かなくてもコントローラーが呼ばれる(!)
 - GET /users → UserController::index()
- さらに後発のフレームワーク(Laravelなど)は、Rails風の便利さは残しつつも明示的に設定させることでコードの追いやすさを意識



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

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



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'
```


PHPについてのFW

- RailsアプリケーションはControllerを書かないと、そもそも実行されない
- PHPについてのフレームワークは、便利ライブラリ集と見分けがつかない
 - あるいは、お行儀よいアプリケーションを書くための拘束具に過ぎない
- Controllerで `setcookie()` や `header()` 関数も動く(!)
 - そんなことをするとユニットテストが書けなくなってしまう！
- PHP自体は依然としてCGI時代と地続き

PHP以外は CGI時代との 分断・断絶を経験

PHPは
FWが一般化後も
断絶を経験してない

ここまでは
過去の栄光

PHPは現在も
パフォーマンス改善
されているが、

頭にはいろいろな言語が浮かぶ

Node.js... C#... ASP.NET...
Rust...
Go... Common Lisp...
Python... Java... TypeScript...
Ruby on Rails...
Kotlin... Swift... Smalltalk...

PHPよりもっと
速い言語で書いた方が
いいじゃん

PHPより
エレガントな言語で
再構築すべきだ

常にさまざまな
誘惑がある



Why Engineers Can't Be Rational About Programming Languages



Steve Francia

3 Nov 2025

Programming-Languages

Leadership

Engineering-Culture

Management

Economy

キャリアの初期、私は前途有望なソーシャルネットワークであるTakkleで働いていました。突然の退職により、リードエンジニアからエンジニアリング担当副社長に昇進し、12名のチームを率いることになりました。私たちはすべての目標を達成していましたが、私は20代前半で経験不足でした。取締役会はこれを解決したいと考えていました。彼らはCEOに、より経験豊富なCTOを採用するよう圧力をかけました。私は彼から学ぶことを楽しみにしていました。彼はPerlコミュニティではよく知られた人物であり、O'Reillyの「キャメル」本を山ほど持ってやって来ました。

彼が最初に行ったことの一つは、私たちの言語であるPHPの選択が間違っていると宣言することでした。彼はPerlへの移行を命じました。この命令は、私にはPHPとPerlを比較した偽りの分析のように感じられるものの後に下されました

私たちの速度は崩壊しました。チームは新しい言語を学ぶだけでなく、ゼロから再構築しなければならず、製品の発売が9か月遅れました。Perlベースの新しいシステムを構築しながら失われた速度を補うために規模を2倍以上に拡大したため、月間バーンレートは20万ドルから50万ドルに跳ね上がり、ランウェイは半減しました。

コードをエレガントに
書き直すのは
今やるべきなのか？

高速な言語で
書き直すのは
今やるべきなのか？

リソース配分



PHPはどこでも
65点を出せる言語

…と言いたいが、
PHPにも歯が
立たない局面もある

C10K Problem

チャットアプリで
結果をリアルタイムに返す

WebSocket Server-Sent Events

preforkモデル
(mod_php, PHP-FPM)は
相性最悪

PHPはそのまま
白旗を上げる
しかないのか

FRANKENPHP

Goで書かれた、モダンな
PHPアプリサーバー

始める



勝った！
PHP大勝利

.....とは行かない

グローバル変数や
静的プロパティが
リセットされない

CGIとは何か

- Common Gateway Interfaceの略、1993年頃に登場
- HTTPサーバを独自実装しなくても、シンプルにスクリプトを書くだけで動的ページを実現できる
- ユーザーからリクエストが来る度にプログラムを起動する
- シェアードナッシング（並行して処理するリクエストと干渉しない）

この環境を全部
捨てることになる

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)

問題になつていない
ことを悩んでも
仕方ない

本当にPHPに
不得意な仕事をさせる
必要があるのか？

本質的な問題を
解決できる
言語を使おう