

# Language Update **PHP** 2018

<?php

2018-08-26 Learn Languages 2018  
#ll2018

# ！ お前誰よ



- うさみけんた (@tadsan) / Zonu.EXE
  - GitHub/Packagistでは id: zonuexe
- ピクシブ株式会社 pixiv運営本部
- Emacs Lisper, PHPer
  - Emacs PHP Modeのメンテナ引き継ぎました
  - 好きなリスプはEmacs Lispです
- Qiitaに記事を書いたり変なコメントしてるよ

# | 言語的なバックグラウンド

- 2004年頃: ISO-HTML
- 2006年頃: JavaScript
- 2010年頃: Ruby, 魔法言語 リリカル☆Lisp
- 2011年頃: Python, shell(Bash), F#/OCaml
- 2013年頃: Emacs Lisp, PHP

基本はWebから始まって動的言語やってきた

<?php

みんな大好きPHP

ですよね？

(暗黒微笑)

リアルワールドでのシェア  
の割にプログラミング言語  
として興味関心の対象と  
してゐる人は多くない印象





*“We have things like protected properties.  
We have abstract methods. We have all  
this stuff that your computer science  
teacher told you you should be using.  
I don't care about this crap at all.”*

*–Rasmus Lerdorf*

[https://en.wikiquote.org/wiki/Rasmus\\_Lerdorf](https://en.wikiquote.org/wiki/Rasmus_Lerdorf)

“PHPにはprotectedプロパティも抽象メソッドもありますよ。計算機科学の教授が「使え」と言ってるものは全部。そんなことは~~クソ~~興味ないですけど”

—Rasmus Lerdorf

原作者のRasmusはプログラミング言語の設計にはあまり関心がない

おしながき

1. PHPの開発体制
2. 言語としての特徴
3. 最近の動向と「次」

# PHPの 開発体制

Rasmusは「最初の開発者」  
であってGuidoやMatzの  
ようなBDFLではない

(Wikipediaの一覧には載ってるけど…)

機能追加や改廃はRFCに  
まとめられMLで議論され  
た後にコミッターによる  
投票で決まるプロセス

<https://wiki.php.net/rfc/voting>



PHPの行く末を  
知りたければRFCを  
ウォッチすればよい

## Request for Comments

---

This page gives an overview of the current RFCs for PHP.

To create a new RFC, see [How To Create an RFC](#).

Note: An RFC is effectively “owned” by the person that created it. If you want to make changes, get permission from the creator. If no agreement can be found, the only course of action is to create a competing RFC. In this case, the old RFC page should be modified to become an intermediate page that points to all the competing RFC's.

A new page in this RFC namespace will automatically be loaded with an RFC [template](#). Customize as needed.

### In voting phase

---

### Under Discussion

---



<https://wiki.php.net/rfc>

# php RFC Watch

[Notify me when the vote for RFC is completed](#)

[array\\_key\\_first\(\), array\\_key\\_last\(\) and array\\_value\\_first\(\), array\\_value\\_last\(\)](#)

Add array\_value\_first() and array\_value\_last()?



Votes cast: 33      Yes: 15 (45%)      No: 18 (55%)

Add array\_key\_first() and array\_key\_last()?



Votes cast: 32      Yes: 18 (56%)      No: 14 (44%)

[Deprecations for PHP 7.3](#)

Deprecate (and subsequently remove) pdo\_odbc.db2\_instance\_name php.ini directive?

[iterable to array\(\) and iterable\\_count\(\)](#)

Add iterable\_count()?



Votes cast: 37      Yes: 2 (5%)      No: 35 (95%)

Add iterable\_to\_array()?



Votes cast: 32      Yes: 8 (25%)      No: 24 (75%)

[User-defined object comparison](#)

Support user-defined object comparison?



Votes cast: 17      Yes: 5 (29%)      No: 12 (71%)

<https://php-rfc-watch.beberlei.de/>

PHP RFC Bot



RFC Bot

ツイート  
406フォロー  
1フォロワー  
1,020[フォローする](#)

## PHP RFC Bot

@PHPRFCBot

Keep track of new and updated  
[@php\\_net](#) RFCs. Managed by  
[@michaeldyrynda](#). Also check out [php-rfc-watch.beberlei.de](#) for voting status.

📍 Adelaide, South Australia

📅 2016年5月に登録

## Twitterを使ってみよう

登録してあなただけのタイムラインを作  
りましょう

[アカウント作成](#)

ツイート

ツイートと返信



PHP RFC Bot @PHPRFCBot · 8月2日

#PHP RFC moved to Implemented PHP 7.3: Same Site Cookie  
[wiki.php.net/rfc/same-site-...](#)



10



16



PHP RFC Bot @PHPRFCBot · 7月21日

#PHP RFC moved to Implemented PHP 7.3: Deprecations for PHP 7.3  
[wiki.php.net/rfc/deprecatio...](#)



14



9



PHP RFC Bot @PHPRFCBot · 7月18日

New #PHP Under Discussion RFC: Namespace Visiblity for Class, Interface and  
Trait [wiki.php.net/rfc/namespace-...](#)



1



12



18



PHP RFC Bot @PHPRFCBot · 7月17日

#PHP RFC moved to Declined: iterable to `arrav()` and `iterable count()`

## #externals

# Opening PHP's #internals to the outside.

Search with  algolia**Status of ci.qa.php.net?**

🕒 30 days ago 👤 Christoph M. Becker 💬 6

**object destruction php 5.6 vs. 7.2**

🕒 16 days ago 👤 Nicolai Scheer 💬 2

**default\_encoding RFC – aftermath**

<https://wiki.php.net/rfc>

# 言語としての 特徴

Webアプリを最も  
ゆるふわ簡単に  
構築できる言語

昔ながらのCGIと同じく  
レンタルサーバー借りて  
ファイルを置けば設置完了



現在ではレン鯖にFTPで転送とか  
よりHerokuにgit pushした方が  
ずっと簡単じゃね？

と僕も思ってるけど、PHPが利用  
される界隈的にそこまで主流の  
選択肢とはされない感じがある

PHPは様々な  
要素を併せ持つ

1. HTMLテンプレート
2. PerlっぽいCGI
3. Javaっぽいクラス

“PHPにはprotectedプロパティも抽象メソッドもありますよ。計算機科学の教授が「使え」と言ってるものは全部。そんなことは~~クソ~~興味ないですけど”

—Rasmus Lerdorf

ほかの言語にあるよ  
うな機能は概ねある

~~計算機科学の教授が使えと言ってるものは全部~~

全部とは言っていない↑↑↑

抽象クラス, トレイト, インターフェイス, ジェネレータ, イテレータ, クロージャ, 関数の動的呼び出し, 無名クラス, eval, 型宣言, 連想配列, リフレクションAPI, 自己文書化, 標準入出力の読み書き, REPL, クラスの遅延ロード

# イテレータ・ジェネレータ

---

- foreachでイテレーションできるモノをユーザー定義で作れる
- ストリーム(遅延リスト・無限リスト)も当然扱える
- 創意工夫でかなりの使い度がある

# トレイト (trait)

---

- 継承なしでクラスに実装追加できる
- 複数トレイトの衝突を解決できる
- 優先順位の定義・メソッド名変更
- つまり「本物の」トレイトの実装
  - Nathanael Schärli, Stéphane Ducasse, Oscar Nierstrasz, Andrew P. Black(2002)  
Traits: Composable Units of Behaviour



# ！ クラスの遅延ロード

---

- 実行時に未定義クラスが登場したときにファイルを読み込むロジックをユーザー定義できる
- いちいちrequire\_onceを書かない
- 不必要なファイルが読み込まれない
- AOP ~~メタプログラミング~~の温床
- Composerを利用するのが鉄板構成

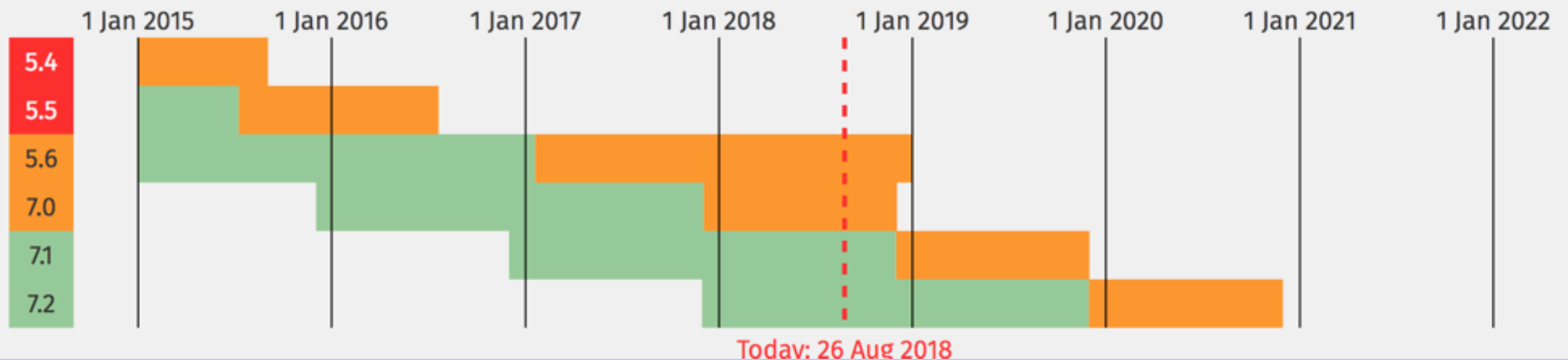
最近の動向

PHP7.0は12月1日前後に  
リリースされるスケジュー  
ルで概ね固定されてる

## Currently Supported Versions

| Branch                | Initial Release |                        | Active Support Until |                      | Security Support Until |                      |
|-----------------------|-----------------|------------------------|----------------------|----------------------|------------------------|----------------------|
| <a href="#">5.6</a> * | 28 Aug 2014     | 3 years, 11 months ago | 19 Jan 2017          | 1 year, 7 months ago | 31 Dec 2018            | in 4 months          |
| <a href="#">7.0</a>   | 3 Dec 2015      | 2 years, 8 months ago  | 3 Dec 2017           | 8 months ago         | 3 Dec 2018             | in 3 months          |
| <a href="#">7.1</a>   | 1 Dec 2016      | 1 year, 8 months ago   | 1 Dec 2018           | in 3 months          | 1 Dec 2019             | in 1 year, 3 months  |
| <a href="#">7.2</a>   | 30 Nov 2017     | 8 months ago           | 30 Nov 2019          | in 1 year, 3 months  | 30 Nov 2020            | in 2 years, 3 months |

Or, visualised as a calendar:



<http://php.net/supported-versions.php>

2年間のメンテと  
1年間のセキュリティ修正

大事な  
お知らせです

2004年から長らくご愛顧  
いただきましたPHP5.xの公式  
サポートは2018年をもって  
完全に終了いたします。  
(5.6のSecurity fixes終了)

ついでにPHP7.0の  
Security fixesも12月3日に  
終了するので7.1または7.2系  
にアップデートしましょう。



あくまでThe PHP Groupからのリリースの話なので、脆弱性が出たら各ディストリ独自のアップデートがあると思いますが。




ご安全に。

# 最近の動向

## (言語コア)

PHP 5.6 → 7.0

これは内部構造の  
最適化と型宣言の  
追加が目玉

.....  
2016年8月28日（日）    Language Update PHP編（LLoT補足）     [22 users](#)      ★

昨日8/27にLLoTの「Language Update」の10分枠でPHPの話をしました。発表資料は以下です。



<http://d.hatena.ne.jp/hnw/20160828>

PHP 7.1以降

微妙に書きにくかった  
文法の修正と危険な機能  
の非推奨化(削除)が中心



# | PHP 7.1

---

- nullableの追加
- 関数戻り値にvoidが書けるように
- 配列分解の短縮構文
- クラス定数に可視性を定義可能に
  - private const

# | PHP 7.2

---

- 型宣言にobjectが書けるように
- 継承時にメソッドの型宣言を省略可
- クラスのグループ表記ケツカンマ可

# PHP 7.3 (12月リリース予定)

---

- 7月にfeature freezeされ現在beta2
- 関数呼び出し引数のケツカンマ可
- ヒアドキュメント／nowdocの改善
  - 終端行判定がシビアだったのが緩和
  - 行頭空白をトリムできるように

PHP 7.4?

# | PHP 7.4

---

- typed propertyが入る…かも(議論中)
  - これは比較的確度高そう
- Fiberが入る… かも(議論中)
  - PHPで非同期処理は伝統的に鬼門
  - 取り込まれるまでハードル高そう

# | PHP 8?

---

- PHP5時代の負債っぽい仕様は一掃？
  - 臆測です
- JITが入る…？
  - ほんとにうまくいくの？
- ローカル変数の型付け？
  - typed propertyの流れで視野内

# 最近の動向

(関連コミュニティ)

# | PHPカンファレンスは活発

---

- 2018年はPHPコミュニティ・カンファレンスが活発化した一年だった
- iOSDC主宰の長谷川さん(@tomzoh)がPHPカンファレンスと別にPHPer Kaigiを開催してコミュニティに火をつけた
- 各地域のCFP応募数が倍増した



# | 2018年のPHPカンファレンス

---

- 2019年は既に例年以上の開催告知
  - PHPerKaigi 2019
  - 地域: 福岡 / 仙台 (New!) / 札幌
  - フレームワーク: CakePHP, Laravel

# | PhpStormの普及

---

- JetBrains社のPHP(+WEB)専用IDE
- 静的解析機能がめっちゃ※強い(PHP比)
  - 入力補完やエラー表示が強い
  - 簡単な型推論はしてくれる
  - PHPDocの型記述を読んでもくれる
- リファクタリング機能がある

# | PHPと静的解析

---

- PhpStorm登場以前は単純なコードのスタイルや複雑度を検査するものが多かった
- PhpStorm以後は変数のスコープや静的な型検査をベースにチェックできるツールが増えた
  - Phan, PHPStan, Psalm

# | PHPDocの現状

---

- ポピュラーなライブラリやフレームワークはPHPDocが整備されてるので、PhpStormを入れるだけで十分補完が機能する状態
- 一方で仕様の標準化と実装は難航
  - PhpStormがPSR-5入れてくれない
  - IntersectionType(非標準)入りそう

# | 並列処理・非同期処理

---

- PHPでは実行モデル的に鬼門  
↑ 情報科学の教授に怒られそう
- それを解決しようとする動きがある
  - Swoole <https://www.swoole.com/>
  - Fiber(RFC) <https://wiki.php.net/rfc/fiber>
  - PHP-Promise(ext)  
<https://github.com/php-promise/promise>

# 書籍

---

- あんまり景気よくなさそう
- 初心者向けの入門書は結構出てる
- 最新のモダンな開発事情を伝えるものはずっと出てない
  - パーフェクトPHPは2014年
- いまがんばって企画書を書いてます

まとめ

PHPの活用領域  
は多様



言語としては従来の  
「負債」を解消する  
方向の変更が多い

PHPを継続的に静的解析できる  
環境が既に揃ってるので既存プ  
ロジェクトの改善や新規開発の  
効率化の大きな武器になる