PHPの「歴史的な理由」ってなんだ！？

自己紹介

所属

株式会社TechBowl

住んでるところ

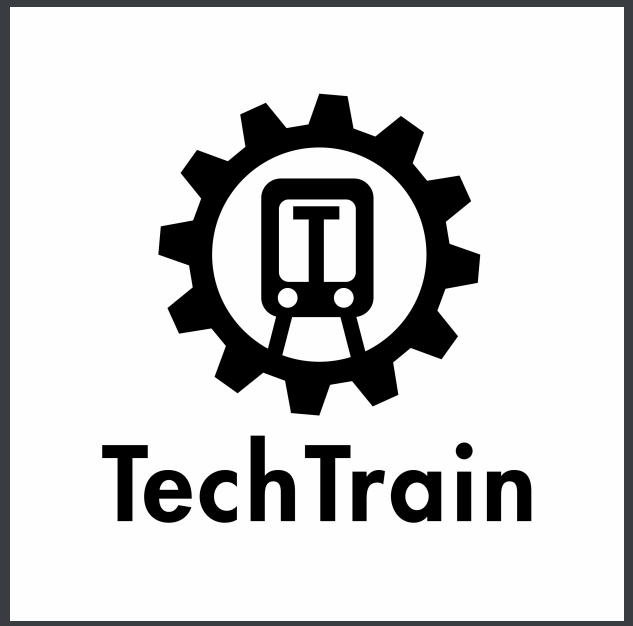
東京

何やってる?

「TechTrain」というサービスで反復横跳びし続けている何でも屋さん(Laravel, Next.js, AWS, etc...)

趣味

- お酒(よく溺れる)
- サウナ
- 読書



TechTrain

エンジニア教育+Directスカウトのサービス。
Coding Stoicをテーマに「うるせえコードかけ！」と言いがちなメンターが多めのエンジニアを育てるためのサービスです。

なぜ発表しようと思ったのか?

TechTrain

PHPの言語自体にディープダイブしていきたい

歴史を遡っていけば、いいんじゃないか

言語の経緯を知った方がより良いコードが書けるのではないか

さて!

PHPの「歴史的な理由」ってなんだ！？

今回初めてCFPを出させてもらったわけですが、、、

PHPカンファレンス関西2024

採択 2024/02/11 15:10〜 $room['C'] レギュラートーク（15分） 初登壇

PHPの「歴史的な理由」ってなんだ！？

スー suguru_ohki

☆ 7

PHPは20年を超える歴史を持つ言語で、php.netでは度々「歴史的な理由」という単語を見かけます。
このトークでは、php.netの標準関数における説明において「歴史的な理由」がなぜ生まれたのかを追い、PHPの歴史とともにその理由について深掘ります。


歴史的な理由 PHP
画像
動画
ショッピング
ニュース
書籍
地図
フライト
ファイナンス
About 51,400,000 results (0.45 seconds)
SlideShare
https://www.slideshare.net › ebihara · Translate this page
お前は PHP の歴史的な理由の数を覚えているのか
Mar 14, 2014 — Kousuke Ebihara. PHP といえば印象的なのは「歴史的な理由」 (≒黒歴史) の
数々ですね。 このセッションでは、普段闇にこもっていてスポットの当たることの少ない「...

Save slide

TechTrain

お前は PHP の歴史的な理由の数を覚えているのか

Kousuke Ebihara (海老原昂輔)

<kousuke@co3k.org>

過去の発表がすでにあった・・・

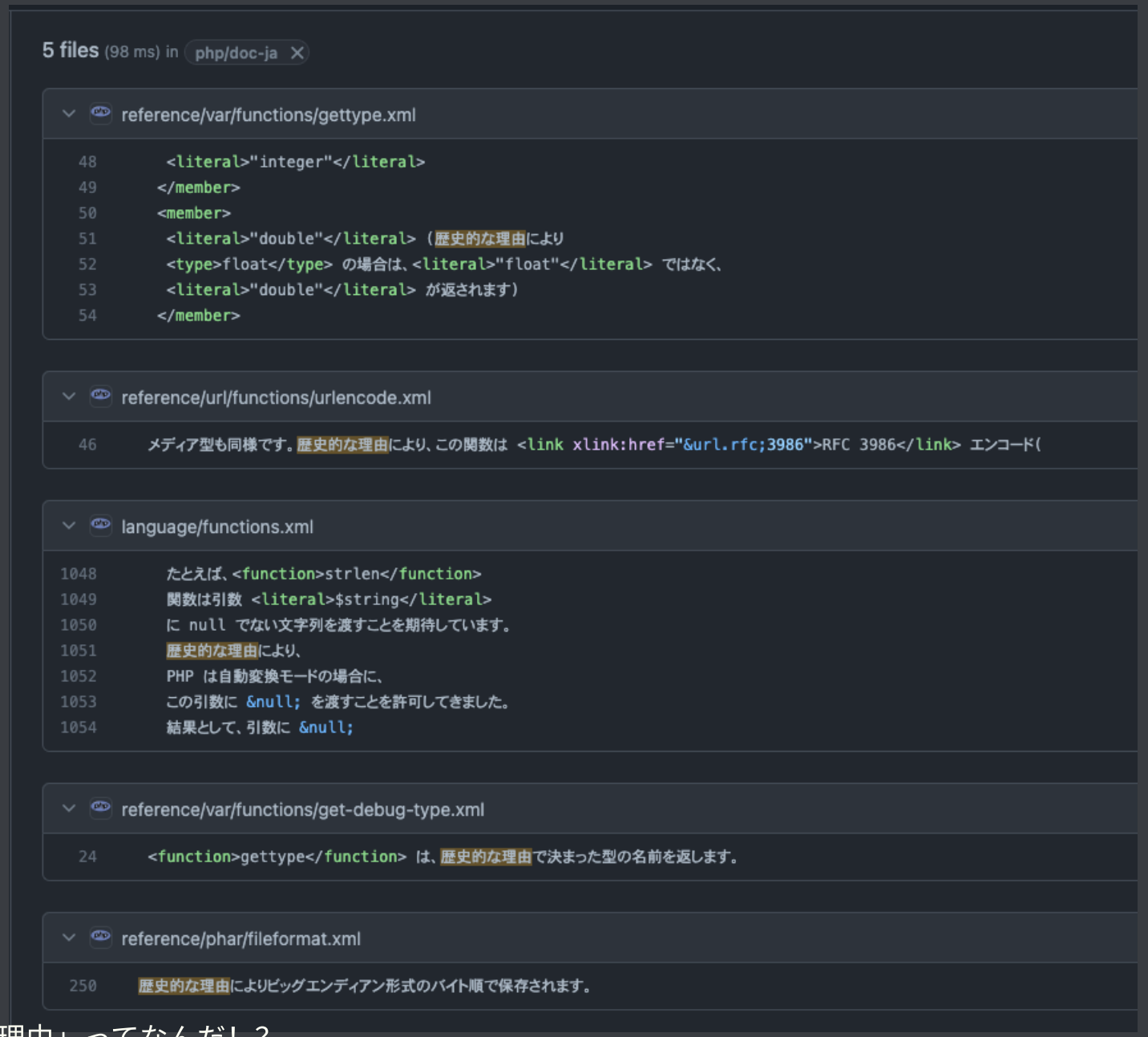
確認してから応募しろ・・・

ここで解説されているものは何か?

歴史的な理由のある機能

- implode()
- urlencode() / rawurlencode()
- double 型 / float 型と、gettype() の返り値
- Phar アーカイブのマニフェスト情報
- Zend Engine の HashTable (間に合わず)

現在のphp.netにおける「歴史的な理由」の数は?


5 files (98 ms) in php/doc-ja
reference/var/functions/gettype.xml
48 <literal>"integer"</literal>
49 </member>
50 <member>
51 <literal>"double"</literal> (歴史的な理由により
52 <type>float</type> の場合は、<literal>"float"</literal> ではなく、
53 <literal>"double"</literal> が返されます)
54 </member>
reference/url/functions/urlencode.xml
46 メディア型も同様です。歴史的な理由により、この関数は <link xlink:href="&url.rfc;3986">RFC 3986</link> エンコード(
language/functions.xml
1048 たとえば、<function>strlen</function>
1049 関数は引数 <literal>$string</literal>
1050 に null でない文字列を渡すことを期待しています。
1051 歴史的な理由により、
1052 PHP は自動変換モードの場合に、
1053 この引数に &null; を渡すことを許可してきました。
1054 結果として、引数に &null;
reference/var/functions/get-debug-type.xml
24 <function>gettype</function> は、歴史的な理由で決まった型の名前を返します。
reference/phar/fileformat.xml
250 歴史的な理由によりビッグエンディアン形式のバイト順で保存されます。

5つ！

過去の発表と被ってないものは実質1つだけ！？ (ユーザー定義関数)

他にないのか・・・！？

GitHubで再度「英語で」検索をしてみました

TechTrain

ん?

reference/snmp/functions/snmpwalkoid.xml

```
25
26    <para>
27     The existence of <function>snmpwalkoid</function> and
28     <function>snmpwalk</function> has historical reasons.  Both
29     functions are provided for backward compatibility.
30     Use <function>snmprealwalk</function> instead.
31    </para>
```

日本語ドキュメントの検索では引っかからないやつ!

TechTrain

一安心ということで、次の2つを遡っていきます！

1. ユーザー定義関数

2. snmpwalkoid

どうやって歴史を追うのか？について

歴史を追うのに利用する情報源

1. PHPのRFCの履歴(PHP5.3~有効)を読む
2. php-srcのGitHubのソースコードのcommitを遡って見る(PHP4.0~有効)
3. PHP Internals(PHP3.0~から有効)
4. museum.php.netでphp-srcのソースコードをダウンロードして見る(PHP1~)

それでは早速追っていきましょう。

1. 内部（ビルトイン）関数

注意:
ビルトイン関数のスカラー型の引数には、 デフォルトの自動変換(coercive)モードの場合、 null を渡すことが出来ます。 PHP 8.1.0 以降では、 nullable として宣言されていない内部関数の引数に **null** を渡すことは推奨されなくなり、 自動変換モードでは警告が発生するようになっています。 ユーザ定義の関数においては、スカラー型の引数は nullable と明示的にマークする必要があり、その振る舞いと合わせるためです。

たとえば、strlen() 関数は引数 $string に null でない文字列を渡すことを期待しています。歴史的な理由により、PHP は自動変換モードの場合に、 この引数に **null** を渡すことを許可してきました。 結果として、引数に **null** を渡すと暗黙のうちに文字列にキャストされ、 結果は空文字列 "" になっていました。 これに対し、strict モードの場合は TypeError がスローされます。

```php
<?php
var_dump(strlen(null));
// "Deprecated: Passing null to parameter #1 ($string) of type string is deprecated" as of PHP 8.1.0
// int(0)

var_dump(str_contains("foobar", null));
// "Deprecated: Passing null to parameter #2 ($needle) of type string is deprecated" as of PHP 8.1.0
// bool(true)
?>
```

> Historically, the reason for this discrepancy is that internal functions have supported a concept of scalar types (bool, int, float, string) long before they were introduced for user-defined functions in PHP 7.0, and the existing implementation silently accepted null values. For the new scalar type declarations introduced in PHP 7.0 an explicit choice was made to not accept null values to non-nullable arguments, but changing the existing behavior of internal functions would have been too disruptive at the time.

PHP RFC: Deprecate passing null to non-nullable arguments of internal functions

TechTrain

> 歴史的に見ると、この不一致の原因は、内部関数が PHP 7.0 でユーザー定義関数に導入されるずっと前から スカラー型 (bool, int, float, string) の概念をサポートしており、 既存の実装では null 値を黙って受け取っていたからです。PHP 7.0 で導入された新しいスカラー型の宣言では、null 値を受け取らないようにすることが明示的に選択されましたが、 内部関数の既存の振る舞いを変更することは、その時点ではあまりに破壊的でした。この RFC では、PHP 8.1 で非推奨の警告を発生させ、内部関数の挙動を同期させることを提案しています。

PHP RFC: Deprecate passing null to non-nullable arguments of internal functions

種類	nullに対する対応
ユーザー定義関数	PHP7.0から導入され、nullを受け取る場合とそうではない場合が最初から区別されていた
内部関数	PHP7.0はるか以前からnullを黙って受け取っていた

知っての通り、PHP8.1からは、内部関数もstrictモードの際には、ユーザー定義関数と同じように明示されているnullableの引数以外は、nullを受け取ることができなくなっています。

ちなみにscalarというワードがいつからあるのか? を遡ってみると

List: php-internals
Subject: [PHP-DEV] CVS update: php3
From: zeev <php-dev () php ! iquest ! net>
Date: 1998-05-01 15:38:26
[Download RAW message or body]

Date: Friday May 1, 1998 @ 11:38
Author: zeev

Update of /repository/php3
In directory asf:/tmp/cvs-serv22463

Modified Files:
ChangeLog language-parser.y
Log Message:
Consider exit() as abnormal shutdown. Unfortunately, there's some memory chunks we \
simply can't free explicitly with some exit() cases, so we have to rely on the memory \
manager to do it for us. In these cases, there's no point to report memory leaks.

Right now, there's a chance some 'real' memory leaks will go undetected, but at least \
there won't be any non-existent ones reported.

Index: php3/ChangeLog
diff -c php3/ChangeLog:1.314 php3/ChangeLog:1.315
*** php3/ChangeLog:1.314 Fri May 1 08:26:05 1998
--- php3/ChangeLog Fri May 1 11:38:25 1998
****

1998年!

TechTrain

これは確かに歴史的な理由ですね・・・。

TechTrain

どのバージョンからなぜnullが暗黙的に受け取られていたのか?
については、遡ってみたものの判明せず、迷宮入りでした・・・

力及ばずごめんなさい・・・

それでは2つ目。

2. snmpwalkoid

そもそも見たことがない関数では・・・。

TechTrain

見たことがある方いらっしゃいますかー！

具体的には、何で利用される標準関数なのか?

理解するために色々前提の理解が必要

SNMP

> SNMP（Simple Network Management Protocol）は、UDP/IPベースのネットワーク監視、ネットワーク管理を行うためのプロトコルです。ルーター、スイッチなどのネットワーク機器、WindowsやUNIXサーバーなどの状態監視、リソース監視、パフォーマンス監視、トラフィック監視を行うために使用します。一般的に、サーバーに対しては、CPU使用率、メモリ使用率、ディスク使用率、プロセス監視、Windowsイベントログ監視、Syslog監視を行います。ネットワーク機器に対しては、各ポート上で送受信されたパケット数、エラーパケット数、ポートの状態（up/down）、およびCPU使用率、メモリ使用率などを監視します。ベンダによっては機器固有の管理項目を公開しているものがあり、きめ細かい監視が可能です。

→ https://blogs.manageengine.jp/itom_what_is_snmp/

SNMPについてのもう少し詳細の説明を入れる。

OID（Object IDentifier）

各情報に対して割り当てられる一意な番号。各SNMPコマンドでの情報取得に必要なID

snmpwalk

そもそもsnmpwalkというコマンドが存在する。
指定したOID配下に含まれるすべてのOIDと、値を取得するコマンド。
snmpwalkoidはそのコマンドのOIDのみを取得できるようにしたコマンドと理解できる。つまりネットワーク監視のために利用されるため、Laravelとか一般的なフレームワークではそもそも利用すらされていない。

→ https://qiita.com/Mabuchin/items/d435c0afb4f0ca17ad25

→ https://www.secuavail.com/kb/log-technique/about-snmp/#:~:text=OID（Object IDentifier）：各,割り当てられる一意な番号。

使い方想定

テストから読み取るのが早そう

https://github.com/php/php-src/blob/b06fedb41d560f756dfb5d964b0d36a224e44dc7/ext/snmp/tests/snmprealwalk.phpt#L21

SNMPの標準関数群について

https://www.php.net/manual/ja/intro.snmp.php
↑こちらを読めば分かるとおり、ラッパーの関数群であることがわかる

> SNMP 拡張モジュールは非常にシンプルで使い勝手の良いツール群です。 Simple Network Managment Protocol を使ってリモートデバイスを管理することができます。
> これは Net-SNMP ライブラリのラッパーなので、 基本的な考え方はこのライブラリと同じです。また、Net-SNMP の設定ファイルや環境変数によって PHP の関数の挙動も変わります。
> Net-SNMP についての詳細な情報は » http://www.net-snmp.org/ にあります。

歴史的な経緯

ここからはいよいよ歴史的な経緯に入っていく

いつからある?

snmpwalkoid

(PHP 4, PHP 5, PHP 7, PHP 8)

ree of information about a network entity

TechTrain

PHP4系からだと思うじゃないですか・・・。

PHP3.0.8 ~

php.netのドキュメントにはPHP4からといったように見える書き方がなされているが、これはgitなどを使ったドキュメントとの連携の管理が4系からなされている影響であり、実際には3系から存在している。

該当の歴史的理由が現在の状態になったところを見てみる

[PHP-DEV] CVS update: php3/doc/functions

色々議論があるのですが、まとめると

1. PHP3.0.8へ入れるため、snmprealwalkという関数が実装される(1999-04-03 2:19:43)
2. PHP3.0.8リリース前にMike Jacksonさんからsnmprealwalkというのは、適切な名前ではない。snmpoidwalkかsnmpwalkoidかに変更してはどうかという提案。下位互換性のためにsnmpwalkoidとsnmprealwalkの動作は同じにするよ！ snmpwalkなどについても考慮するよ！ ポリシー的に適切かわからないけどね！(1999-05-21 13:42:07)
3. Rasmus Lerdorfさんからいいね！ その変更進めようぜ！ PHP3.0.8に混ぜてリリースしちゃおう！ という返信がある(1999-05-22 0:16:23)
4. このあと実装が行われた(1999-05-22)

わずか1ヶ月半くらいが歴史的な理由・・・！ｗ

まとめ

種類	歴史的な理由の内容
ユーザー定義関数	内部関数との動作の差分を入れるための対応が要因
snmpwalkoid	1ヶ月半くらいで提案をサクサク入れたことが発端だったようだ・・・w

ご静聴ありがとうございました!

今日は懇親会には1時間弱ほどの参加になるので、懇親会前でもいるうちに話しかけてくれると大喜びします！！！

おまけ

1. PHP3.0.8はmuseum.php.netにzipファイルがない
2. PHP1系はmuseum.php.netにzipファイルが1つしかない
3. PHP Internalsのメーリングリストを遡るのは結構大変

こちらに調査時などの走り書きを入れています


TechTrain