配列、ジェネリクス、PHPで書けない型

公開補訂版

Arrays, Generics, and Types that cannot be type-declared.

pixiv Inc.
USAMI Kenta

pixiv

2021.10.03

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 pixiv事業本部 エンジニア
  - 最近はピクシブ百科事典(dic.pixiv.net)を開発しています（PSR-15!）
- Emacs Lisper, PHPer
  - Emacs PHP Modeを開発しています（2017年-）
- PHPカンファレンス実行委員、PHPerKaigi コアスタッフ

tadsanのあれこれが読める場所

- https://tadsan.fanbox.cc/
- https://scrapbox.io/php/
- https://www.phper.ninja/
- https://zenn.dev/tadsan
- https://qiita.com/tadsan
- https://github.com/bag2php

WEB+DB PRESS総集編

pixiv

Software Design 11月号

この記事は執筆時点で10月2～3日に開催を予定しているPHPカンファレンス2021と連動した短期集中連載の最終回です。つまり、この記事が掲載されたSoftware Design 2021年11月号をみなさんが手にとられるころにはすでに開催されています。カンファレンスで発表されたセッションはYouTubeチャンネル[注1]にアーカイブが公開されているはずですので、興味のある方はぜひご覧ください。筆者はPHPカンファレンス2021で「**配列、ジェネリクス、PHPで書けない型**」と題したトークを発表します。

最終回では近年の動的言語の型検査の動向についての概観に触れたあと、PHPの動向について説明します。

静的型付き(statically typed)の言語と呼びます。事前にデータの種類を固定せずに処理することを**動的型検査**(dynamic type checking)と呼び、実行時にデータに応じた処理をすることが基本のプログラミング言語を**動的プログラミング言語**(dynamic programming language)と呼びます。

メジャーなものでは、C言語、Java、C++、C#、Go、Swift、Rust、Scala、Kotlin、Haskell、OCamlなどは静的型付きの言語です。また、Python、JavaScript、Ruby、シェルスクリプト、Clojureなど Lisp系の言語、Smalltalk、そしてPHPは動的言語と呼ばれます。俗に実行ファイルを生成するものをコンパイル言語、ソースコードを直接実行するものをインタプリタ言語として

pixiv

今回のお題

pixiv

プロポーザル

採択 2021/10/03 10:00〜 Track2 Long session (60 mins)

配列、ジェネリクス、PHPで書けない型 PHP Conference Japan 2021

☆ 6

 うさみけんた tadsan

近年、PHPの機能強化により型宣言だけで安全に書ける範囲が広まっています。

その一方でPHPStanやPsalmといった静的解析ツールはPHPでは表現できない強力な型を提供することでコード品質向上の価値を高めることが可能です。
PhpStormはPHPプログラマに静的型の恩恵をもたらした一方、先述のツールと比べサポートする型について見劣りする点がありました。
ところが最近リリースされた待望の新バージョンである2021.2はこれまで静的解析ツールの専売特許だった型のいくつかがサポートされるようになり、型検査だけでなく入力補完などの恩恵を受けられるようになりました。そこで追加された型のひとつがジェネリクス（総称型）です。

本発表では、PHPの型についての基礎知識（今日からできる安心型付け入門）があることを前提として、PHPにジェネリクスは入るのか？の内容を軸に、PHPの型宣言では現時点で賄えない部分、特に配列の型およびジェネリクスの概念、class-string型の概念、そしてジェネリクスを実際に活用するためのテクニックを説明します。

Discord Channel: #track2-4-php-type

いきなりですが
問題です

pixiv

値をそのまま返す関数

```
function id(mixed $value): mixed {
    return $value;
}
```


pixiv

値をそのまま返す関数

```
$date = new DateTimeImmutable();
$date->diff(
```

PhpStormとかで
メソッド名補完がきく

pixiv

値をそのまま返す関数

```
$date = new DateTimeImmutable();
id($date)->diff(
```

PhpStormとかで
メソッド名補完が
き…かない

pixiv

値をそのまま返す関数

```php
function id(mixed $value): mixed {
    return $value;
}
```

なんでも受け取って

なんでも返す

pixiv

本日の目標は
こういうときに怯えず
型を付けることです

個々の静的解析ツール
導入方法などは
今回は扱いません

ただし最新のPhpStorm 2021.2では静的解析ツールを導入しなくてもすんなり動くと思います

これまでのあらすじ

pixiv

PHPカンファレンス2016

pixiv

PHPカンファレンス福岡2019

pixiv

PHP勉強会@東京 (2020年2月)

pixiv

PHPカンファレンス沖縄2021

pixiv

型とPHPの歴史

- 2004年 PHP 5.0で限定的ながら型宣言(type hinting)が可能に
  - arrayおよびクラス名(インターフェイス)のみ記述可能
- 2012年頃(?) PhpStormがPHPDocベースの静的解析を提供
- 2015年12月 PHP 7.0リリース、スカラー型宣言とstrict_typesが追加
- 2016年頃 Phan, PHPStan, Psalmが開発開始
- 2019年12月 PHP 7.4リリース、プロパティに型宣言が追加

型についておさらい

型とは何か

- 基本的には値の種類のこと
- PHPのデータ型は以下のように分類できる
  - スカラー型: bool / int / float / string
  - 複合型: array / object / callable / iterable / (mixed)
  - 特殊型: null / resource
- クラスとインターフェイスをユーザー定義型として利用できる

PHPは動的言語

- 変数には型宣言がない
  - 基本的にどのような値でも入れることができる
  - 値の型は実行時にいつでも確認できる
    - is_int(), is_bool(), is_array() などの関数
- 関数とプロパティは型宣言できるし、省略もできる
  - パラメータや戻り値の型はReflectionで実行時にいつでも確認できる

型チェックは実行時まで遅延する

```
<?php

$a = [];
$b = 1;
echo $a + $b;
```

$a : array

$b : int

そもそも
計算できない

pixiv

PHPの型宣言の特徴

- 実行時に型を確実にチェックされる
  - スカラー型(bool, int, float, string)を宣言したとき、
    デフォルトではキャストよりも厳密に値を判定した上で型変換を行う
  - それ以外の型は一致しなかった場合にTypeErrorを発生する
- ファイル単位で declare(strict_types=1) と書くと厳密な型モードになる
- 関数の呼び出し側で宣言したモードが反映される

型チェックは実行時まで遅延する

```php
<?php
$toint = fn($n): int => $n;

var_dump($toint('1'));
var_dump($toint((int)''));
var_dump($toint(''));
```

パラメータ
mixed

返り値でintに
変換する

1 : int

0 : int

返り値で厳密な
チェック

pixiv

漸進的型付けで動的言語を型検査する

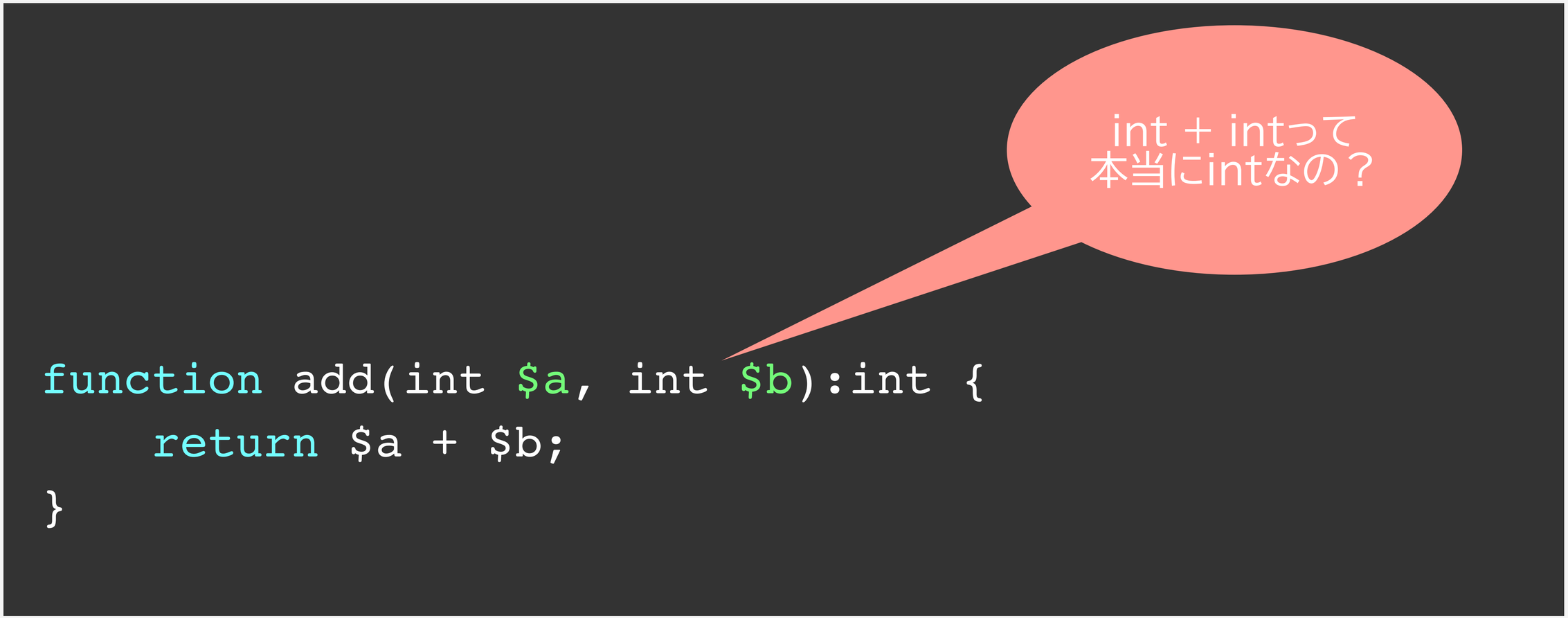
- 何も型宣言されていない状態をmixed型(何でもあり)とする
- より詳細に状態を絞りたいポイントに型宣言やPHPDocなどを付けていく
  - PhpStormやPHPStanなどのツールはPHPDocの内容を考慮してくれる

型がついていない関数

```
function add($a, $b) {
    return $a + $b;
}
```

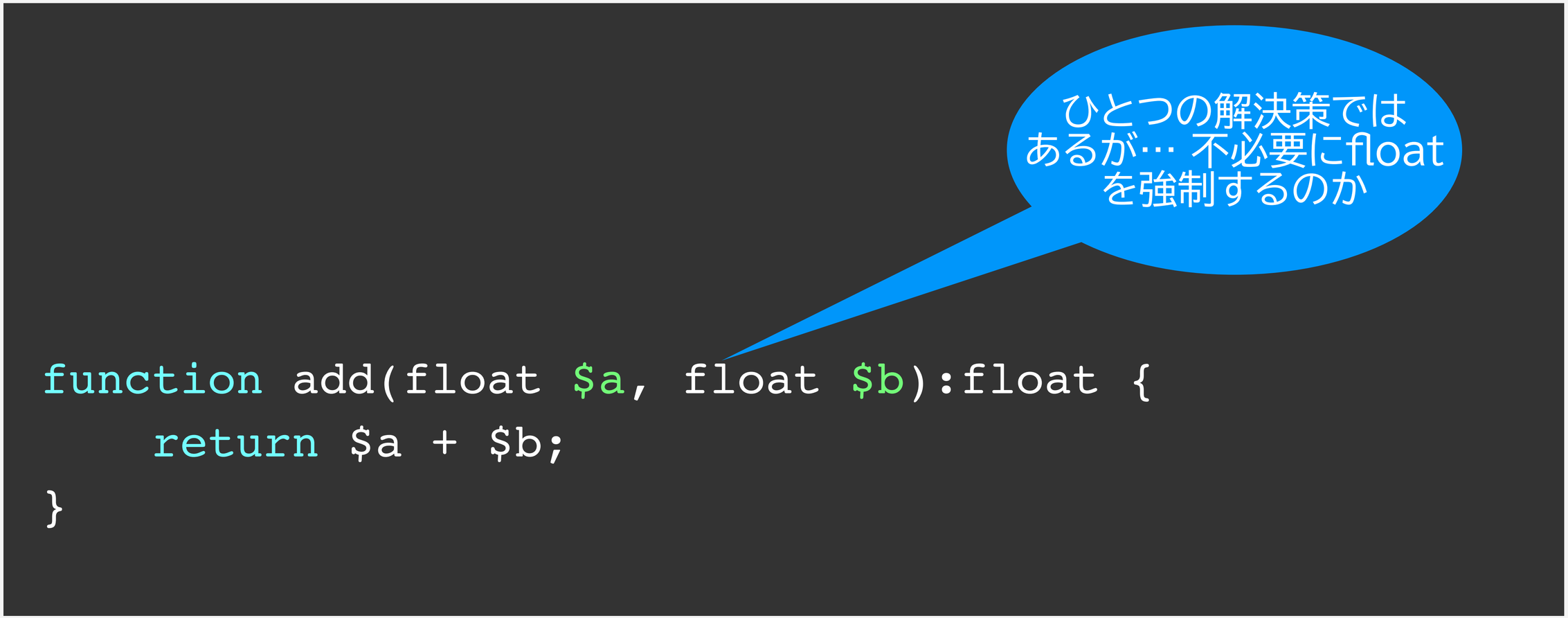

pixiv

型宣言すればいい？



pixiv

floatにすればいい？




```
function add(float $a, float $b):float {
    return $a + $b;
}
```


pixiv

PHPDocの型注釈(アノテーション)

```php
/**
 * @param int|float $a
 * @param int|float $b
 * @return int|float
 */
function add($a, $b) {
    return $a + $b;
}
```

あえて型宣言を省略する

pixiv

Union型宣言(PHP8.0)

```
function add(
    int|float $a,
    int|float $b
): int|float {
    return $a + $b;
}
```


pixiv

かくしてPHPは
PHPDocなんか
書かなくても良い
平和な世界になりました

pixiv

HAPPY END?

pixiv

ほんとうにそうか？

pixiv

Bookクラスは著者を持つ

```
class Book
{
    function __construct(
        private Author $author
    ) {
        // ...
    }
}
```

上司
「共著を想定できてる？」

pixiv

Bookの著者は複数居る

```
class Book
{
    function __construct(
        private array $authors
    ) {
        // ...
    }
}
```

arrayにすることで
複数であることを表現

Authorクラスの
情報が減ってる…

pixiv

著者一覧を出力したい

```
class Book
{
    function printAuthors($fp): void {
        foreach ($this->authors as $author) {
            fwrite($fp, $author->getName());
        }
    }
}
```

Book::$authors : array
中身が何かわからない！

Authorクラスの
メソッドを補完できない

pixiv

Bookの著者は複数居る

```
class Book
{
    function __construct(
        private array $authors
    ) {
        // ...
    }
}
```

arrayにすることで
複数であることを表現

Authorクラスの
情報が減ってる…

pixiv

著者は複数居る

PHPDocの復活

```
class Book
{
    function __construct(
        /** @var array<Author> */
        private array $authors
    ) {
        // ...
    }
```


pixiv

著者一覧を出力したい

```
class Book
{
    function printAuthors($fp): void {
        foreach ($this->authors as $author) {
            fwrite($fp, $author->getName());
        }
    }
}
```

Book::$authors :
array<Author>

Authorクラスの
メソッドが補完できる

pixiv

この事例からわかること

- arrayは独立した型だが、中に何が入っているかはわからない
- array型宣言されていても、どんな配列でも正常に動くわけじゃない
  - むしろ「arrayなら何でも受け付ける」という関数はかなり少数派
  - 「いったいどんな配列なのか」を詳細に説明する必要が生じる
  - 配列の内部詳細を書くための記法がある

配列はひとつ！じゃない！！

- ユーザIDがならんだリスト [123, 456]
- ユーザIDと名前の対応表 [123 => '野比のび太', 234 => '源静香']
- ユーザを表す構造体 ['id' => 123, 'name' => '野比のび太']
- ユーザを表す構造体がならんだリスト

```
[
    ['id' => 123, 'name' => '野比のび太'],
    ['id' => 234, 'name' => '源静香'],
]
```

list<int>

array<int,string>

array{id:int, name:string}

list<array{id:int, name:string}>

pixiv

配列の記法

- 従来は Author[] とか int[] みたいに [] を後置する記法が主流だった
  - この記法では構造体のような配列の内部構造を記述できない
- array<Author> や array<int> で要素を表現できる
- array<int, Author> のように書くとキーも表現できる
- array{id:int, name:string} のような値はarray-shapesと呼ばれる

型パラメータ

array<int>

array<Author>

いろんなものが書ける

再掲: 値をそのまま返す関数

```
function id(mixed $value): mixed {
    return $value;
}
```


pixiv

再掲: 値をそのまま返す関数

```php
/**
 * @template T
 * @phpstan-param T $value
 * @phpstan-return T
 */
function id(mixed $value): mixed {
    return $value;
}
```

値をそのまま返す関数

```
$date = new DateTimeImmutable();
id($date)->diff(
```

PhpStormとかで
メソッド名補完が
きくようになった

pixiv

これを活用する

配列の先頭要素をとる関数

```
function first(array $xs, mixed $default): mixed {
    foreach ($xs as $x) return $x;
    return $default;
}
```


pixiv

配列の先頭要素をとる関数

```
/**
 * @template T
 * @param array<T> $xs
 * @param T $default
 * @return T
 */
function first(array $xs, mixed $default): mixed {
    foreach ($xs as $x) return $x;
    return $default;
```

配列の先頭要素を受け取る

```
// list<DateTimeImmutable> を返す関数
$dates = get_dates();
$first = first($dates, new DateTimeImmutable);
$first->diff(
```

PhpStormとかで
メソッド名補完がきく

pixiv

これを活用する

kane.php

README.md

🤑 kane.php 🤑

PHPで金額計算をするためのクラスです。

> Idol time is money!!

メイクマネー

```php
$十円 = Kane\JPY(10);
$二十円 = Kane\JPY(20);

echo $十円->add($二十円), PHP_EOL;
// "30.0000000000"
```

Kane\Money::eval()

```php
<?php

$moneys = [Kane\JPY(1000), Kane\JPY(2000), Kane\JPY(3000)];
// こう書いても同じ
// $moneys = array_map('Kane\JPY', [1000, 2000, 3000]);
$sum = Kane\Money::eval('+', ...$moneys);
var_dump((string)$sum);
```


pixiv

フォント

- FOT-ハミング Std (M)
- BIZ UDPゴシック（Regular）
- `Courier (Regular)`