なぜPHPに型がつくのか

All Your PHP Code Are Typed by Us

pixiv Inc.
USAMI Kenta

pixiv

2025-12-13 #TechGyoza
小山市立 生涯学習センター ギャラリー

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）

Python初学者向け読書会@札幌 #8

Pythonで学ぶ、初めてのコンピュータサイエンス

日時： 2012/02/07 18:30 to 20:00

☞ Google Mapsで表示

定員： 20 人

会場： **札幌市男女共同参画センター 3F OA研修室 （札幌市北区北8条西3丁目 札幌エルプラザ）**

URL： https://groups.google.com/group/python-sapporo/

管理者： Zonu.EXE, tadsan.

ハッシュタグ： # pysap

Pythonやプログラミングの基礎を学びたいひと向けの読書会を行います。

今回の会場はエルプラザです。ご注意ください。

概要

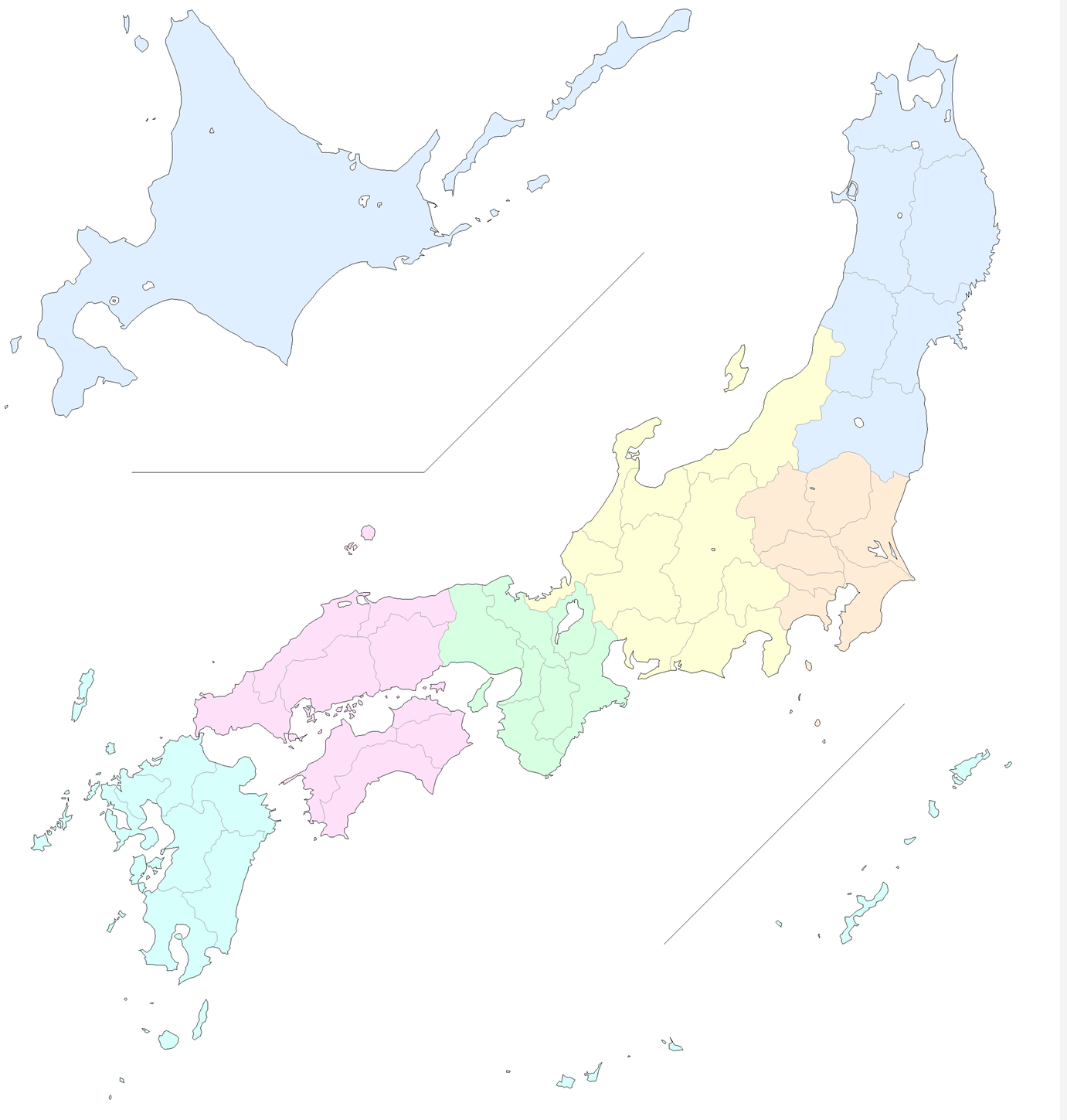
このイベントは終了しました

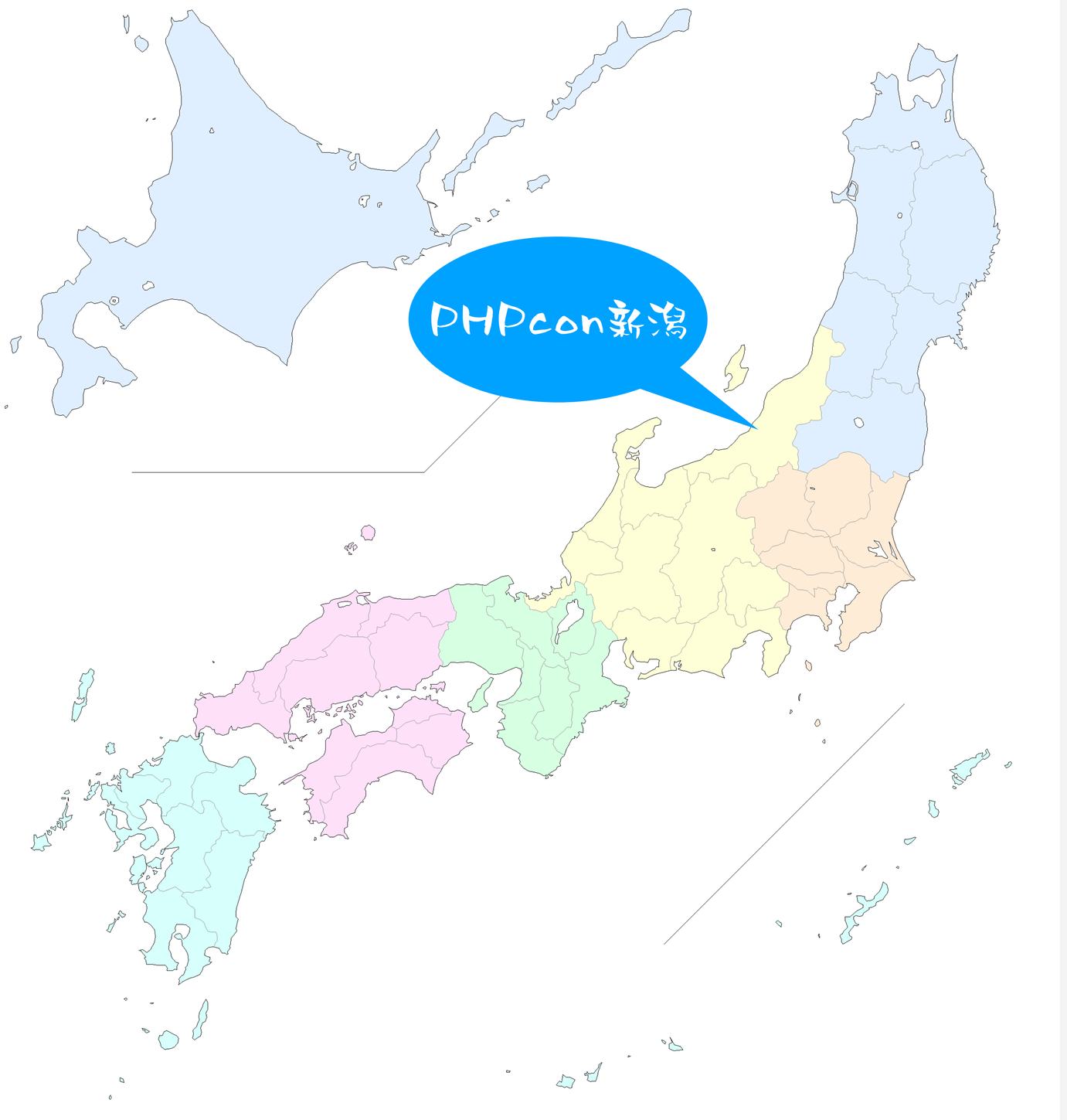
参加希望者 **6** / 20 人

参加：6

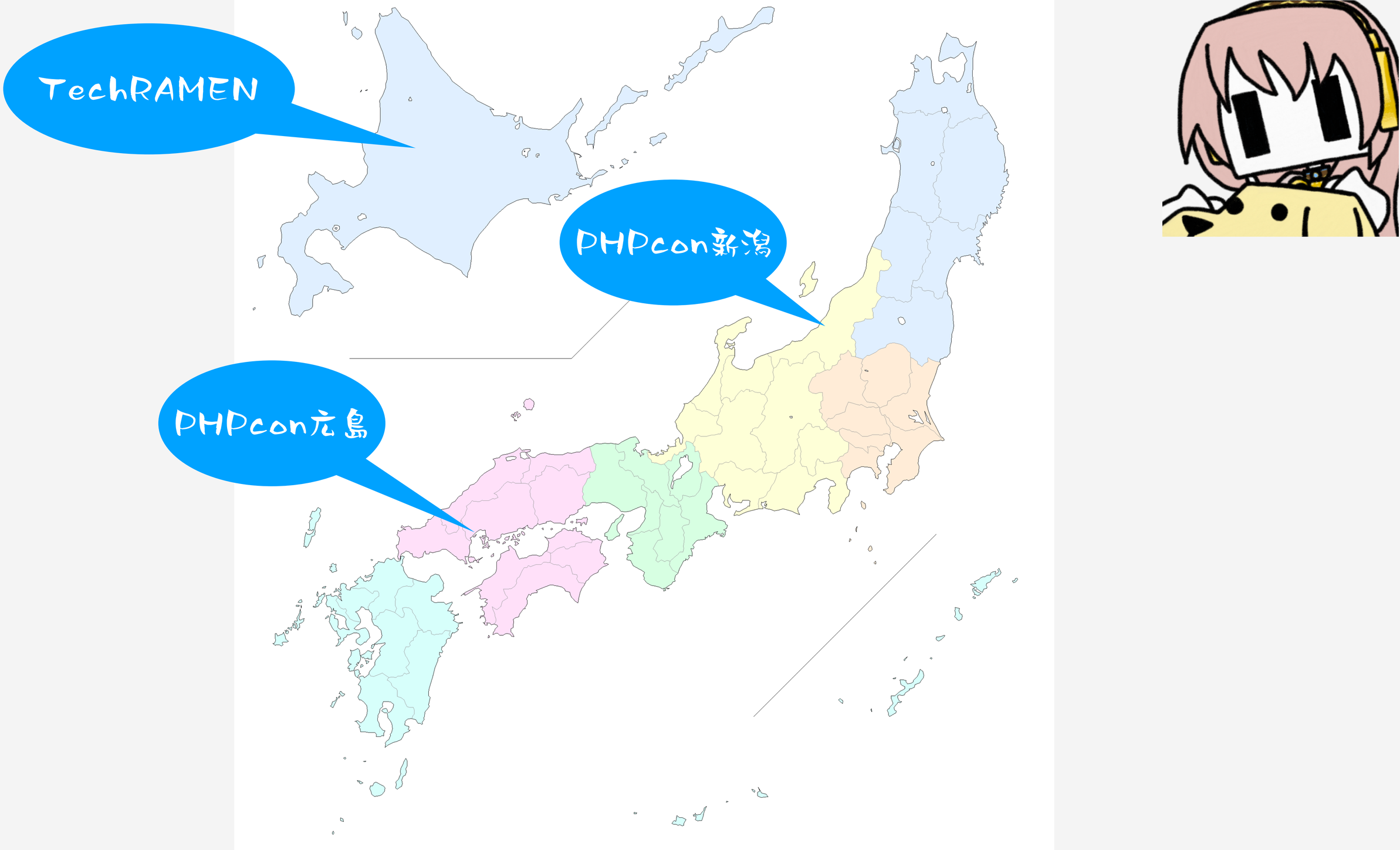
▼ 参加者 (6 人)

1. Zonu.EXE, tadsan. : ﾊｲｻｲ!
2. tunacook : |д゜)ﾁﾗｯ
3. ma2ken_zawa ma2ken_zawa : (゜д゜)ﾊｯ!ｱｹｵﾒ
4. openduck : 〜（､､）〜
5. Aut_Spyke Aut_Spyke : 球春到来
6. arannd : 急にPythonやることになったので参加させて頂きます。




PHPcon新潟


PHPcon新潟
PHPcon広島


TechRAMEN
PHPcon新潟
PHPcon広島

趣味は
プログラミング言語

特に動的言語／スクリプト言語が好き

動的言語には
無限の可能性()が
あります

さて

近年の流れ

2000年代は
動的言語が
大活躍した時代

重厚なIDE…
長大なコンパイル時間…

スクリプト言語は
シンプルなエディタで
簡単に書ける！

2000年前後
CGIで簡単に
Webアプリが書ける

2010年前後
Railsで高機能なものが簡単に書ける

アプリケーションが
大規模に「育つ」と
収拾がつかなくなってくる

メソッドの実装を
わざわざ探して
コードを読まなければ
引数の種類すらわからん

202x年
世界は型の炎に
包まれた！

JavaScript

JavaScript

TypeScript

Python

Python
↓
typing module

Ruby

Ruby

Inline RBS

海は枯れ、地は裂け、
全ての型なしが
死滅したかのように見えた

だが、PHPは
死滅していなかった！

世紀末救世主伝説
<?php

pixiv

誰でも簡単に
ホームページが作れる
ゆるふわ言語

PHPについてのパブリックイメージ

脆弱性 ゆるふわ 弱い型 動的
クソザコ 型なし 意味不明 弱い
自動変換 貧弱 Perlっぽい 適当

PHPについての認識は概ね間違い

~~脆弱性~~ ~~ゆるふわ~~ ~~弱い型~~ 動的

クソザコ ~~型なし~~ 意味不明 ~~弱い~~

自動変換 貧弱 ~~Perlっぽい~~ ~~適当~~

認識を揃えて
おきましょう

動的言語／スクリプト言語

- 実行時の宣言、evalなどプログラム自身を操作対象としながら実行できがち
  - スクリプト／インタプリタなどの特徴は直交する概念だが、結びつきがち
- 関数・変数に静的な型がついてない言語でしょ、という回答は30点くらい
  - C#はいわゆる静的型付きの言語だが、dynamicとして動的な機能も提供

言語としてのPHP

- よくあるC言語風の制御構文と標準関数をもったスクリプト言語
  - if, else, switch, while, for, **goto**
    - PHPのgotoは綺麗なgoto
- 関数定義、クラス定義(class, interface, trait, enum)
- クラスや関数の再定義(オープンクラス・モンキーパッチ)はできない

動的言語でも
型を書くことが
広まってきた

ドキュメント
としての型

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

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

宣言の構文内にも
型を書けるようになった

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

関数に入出力の型を書いておくことで影響を想像しやすく

ここまでは人間が
型を付ける(書く)話

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

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

DocCommentとかいう
机上の空論

pixiv

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

DocCommentとかいう
机上の空論

コーディング時の
ヒントとして役に立つ…
かもしれない

pixiv

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

intしか渡されないことは
実行時に保証される

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


pixiv

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

intしか渡されないことは
実行時に保証される

型宣言に反する値が
返されたら実行時エラー

pixiv

実際には型宣言を
もとに型を再構築する

PHP

```
$a = 1;
$b = 2;
$c = $a + $b;
```


pixiv

PHP

```
$a = 1;
$b = 2;
$c = $a + $b;
```

C言語

```
int a = 1;
int b = 2;
int c = a + b;
```

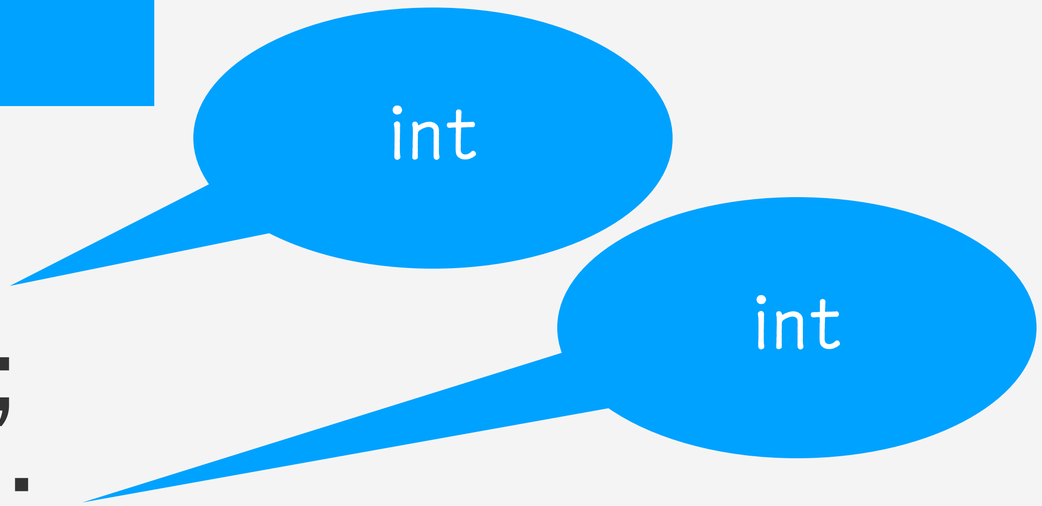
PHP

```
$a = 1;
$b = 2;
$c = $a + $b;
```


pixiv


```
$a = 1;
$b = 2;
$c = $a + $b;
```

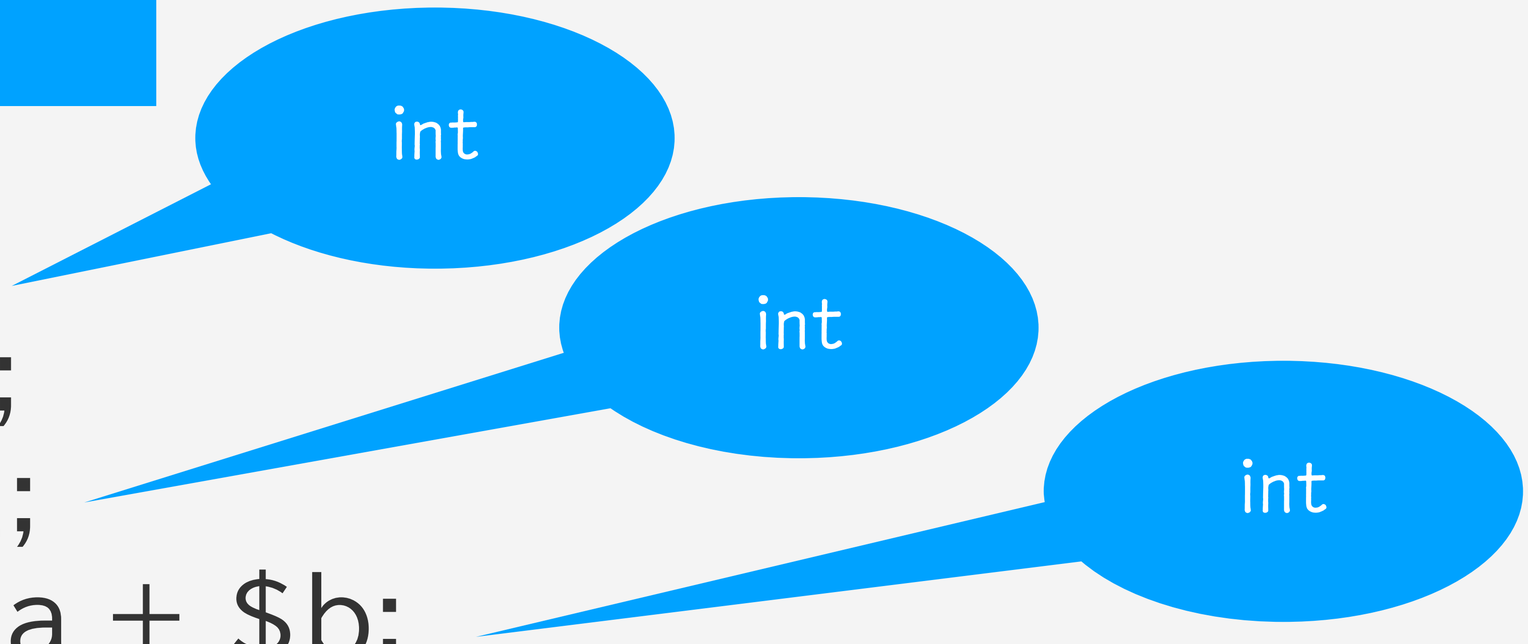
PHP




```
$a = 1;
$b = 2;
$c = $a + $b;
```


pixiv

PHP




```
$a = 1;
$b = 2;
$c = $a + $b;
```


pixiv

PHPでは処理系の外で静的型チェッカーの文明が広がった

型チェッカー

```
$a = 1;
$b = 2;
$c = $a + $b;
```

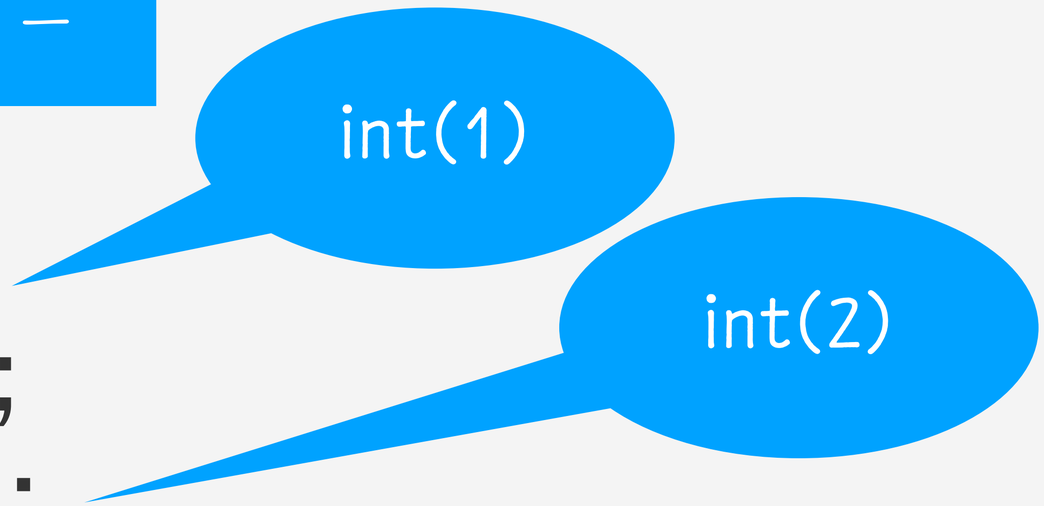
型チェッカー

int(1)

```
$a = 1;
$b = 2;
$c = $a + $b;
```


pixiv

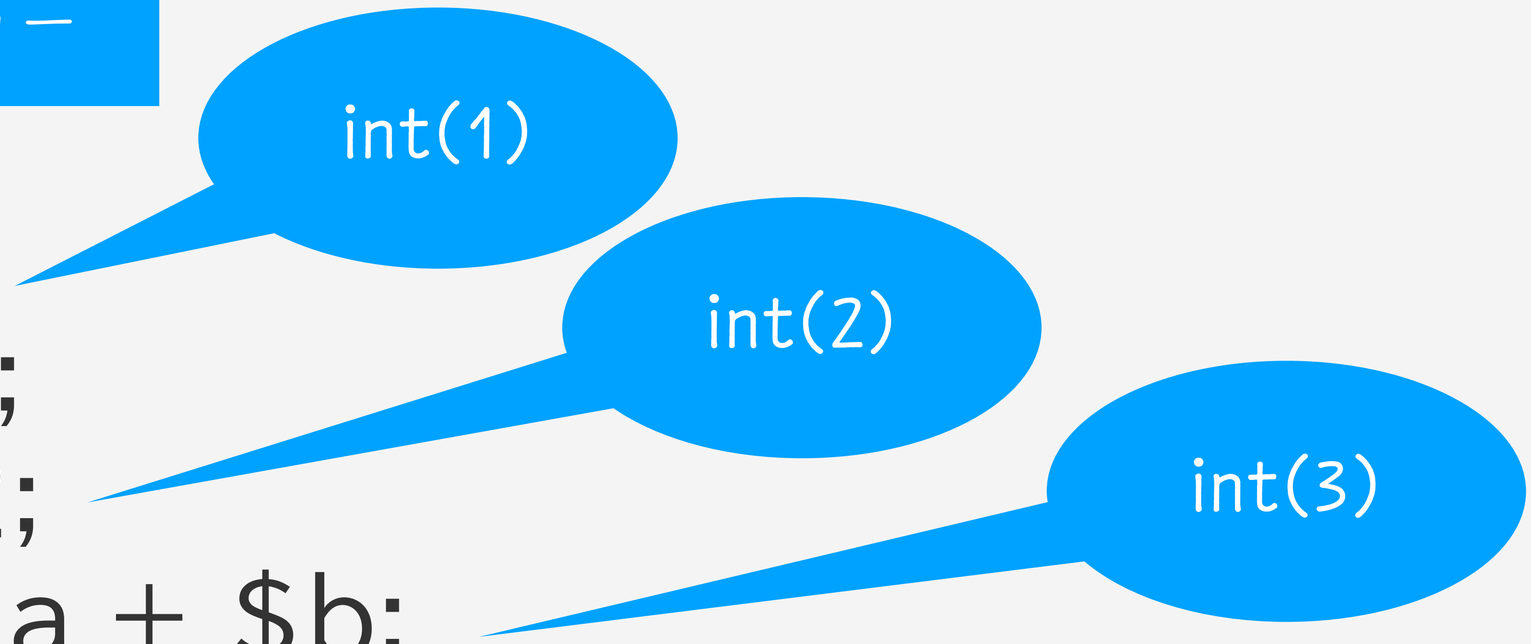
型チェッカー




```
$a = 1;
$b = 2;
$c = $a + $b;
```


pixiv

型チェッカー




```
$a = 1;
$b = 2;
$c = $a + $b;
```


pixiv

型チェッカー

```
$a = ‘foo’;
$b = rand() === 1 ? ‘foo’ : ‘bar’;
$c = $a + $b;
```

型チェッカー


```
$a = ‘foo’;
$b = rand() === 1 ? ‘foo’ : ‘bar’;
$c = $a + $b;
```


pixiv

型チェッカー


```
$a = ‘foo’;
$b = rand() === 1 ? ‘foo’ : ‘bar’;
$c = $a + $b;
```


pixiv

型チェッカー


```
$a = ‘foo’;
$b = rand() === 1 ? ‘foo’ : ‘bar’;
$c = $a + $b;
```

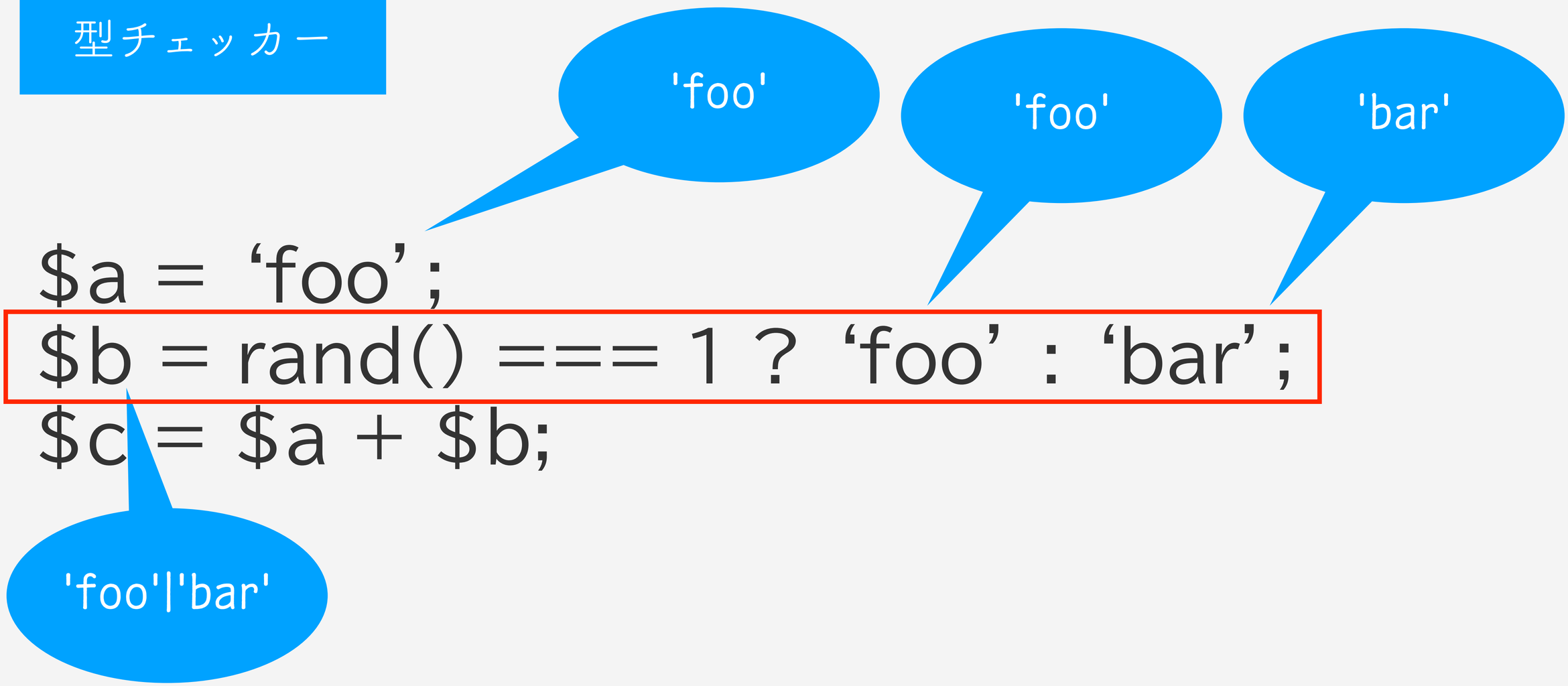

pixiv

型チェッカー


```
$a = ‘foo’;
$b = rand() === 1 ? ‘foo’ : ‘bar’;
$c = $a + $b;
```

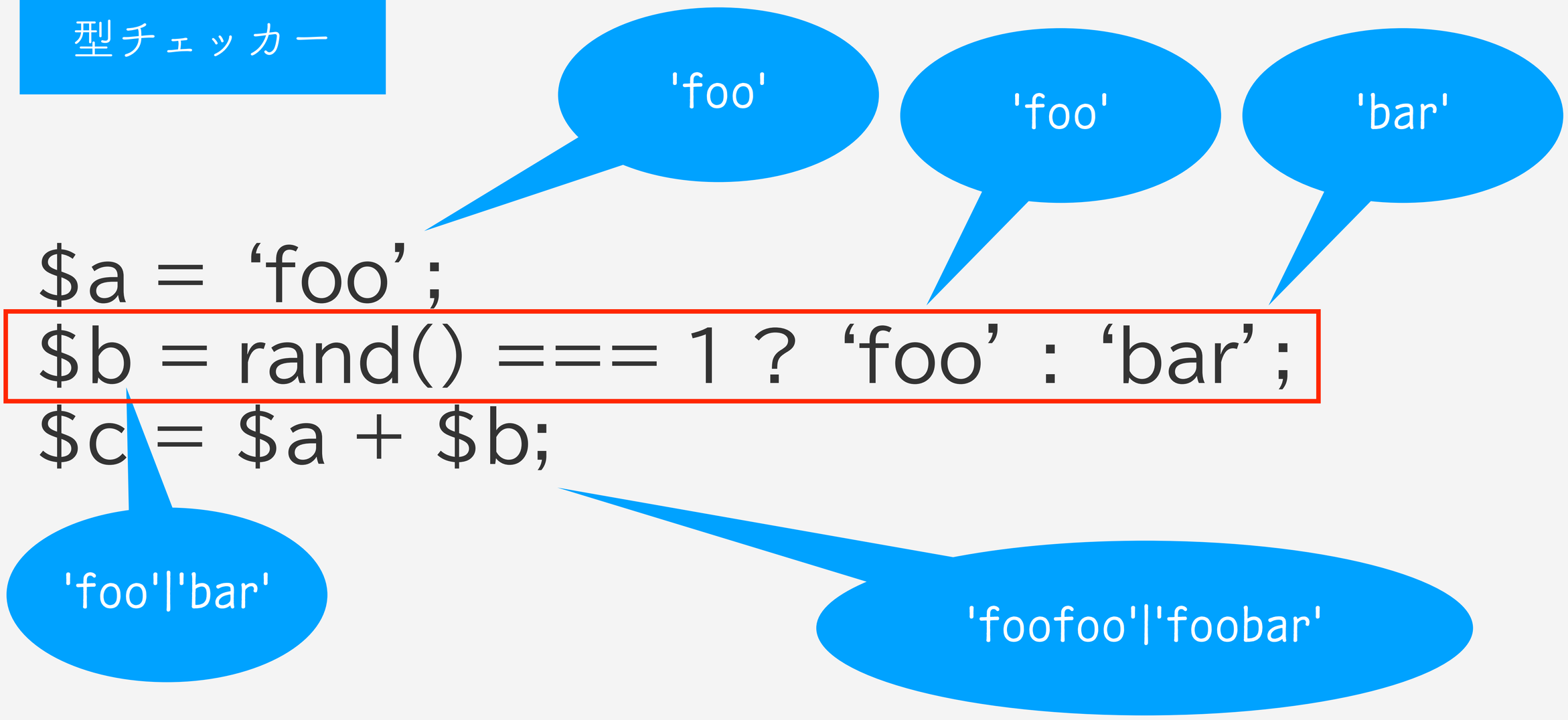

pixiv

型チェッカー



pixiv

型チェッカー



pixiv

いろんなところから湧き出す型の特定できないデータ

```php
<?php

// JSONデータのデコード結果
$data = json_decode($str);

// eval (PHPコードの動的実行)
$data = eval($code);

// DBのクエリ結果
$data = $db->query($sql);

```


49

pixiv

arrayは無限の可能性を秘める(悪い意味で)

```
1  <?php
2
3  function searchUser(int $user_id): array
4  {
5      // ...
6  }
7
8  $user = searchUser(11);
9
10 $user[???] ← どんなキーで格納されてるのか不明
11
```


50

pixiv

扱う対象が複数になったのに型に反映されない

```php
<?php

function searchUsers(array $user_ids): array
{
    // ...
}

$users = searchUsers([11, 22, 33]);

$users[???] ← どんなキーで格納されてるのか不明

```


51

pixiv

PHPの組み込みの型機能ではどんなに型推論しても静的に型が定まらない

無限の可能性を秘めたarray
を放置するとコード全体の
エントロピーが増大し
プロジェクトは熱的死を迎える

TypeScriptにとっての型

- 型宣言をもとに型チェックを行なって、分析情報を報告
- 実行時には型情報は何も残らない
  - JavaScriptにコンパイル ≒ 型定義の部分をひっぺがして実行
    - Node.js 25.2ではType Strippingで直接実行できるようになった
    - 実際にはTypeScript enum など別のコード生成をする機能もある
  - 型がついたコードは実行時にはエラーが出ないでほしい(願望)

PHPにとっての型

- 構文内の型宣言に含まれる型情報は実行時に必ず検証される
  - エラーなく動いているとき、型宣言と実行時の状態が必ず一致する
  - 実行時に検査するのは重そうだが（先入観）実際にはJITで最適化
- PHPDoc（コメント）に書いた型注釈はIDE/静的解析ツールが解釈する
  - PHPDocに書かれたものは机上の空論に過ぎない

動的言語と型

- 自分のプロジェクト内で型を書き始めるのは簡単だが…
  現実には外部のライブラリで型が書かれていないこともある
  - 型が書かれていても実態と一致しないこともよくある
  - Ruby, Python, JavaScriptでは外部プロジェクトで
    ライブラリの型をメンテナンスされているものもある
- PHPでは実行時に強制力のある型宣言があるので、
  実装と一致した型がきちんと書かれていることが多い

そんなに動的言語を書きたいか

書きたい！

静的型付きと動的の
良いとこどりができる
ゆるふわ言語PHPに
これからもご期待ください