快速了解  大進化

抓穩了！要起飛了！

Miles 2020/7/25

About Miles

CURRENT

Senior Developer @ 104 Corp.

TAG

, Laravel, Docker

GitHub

MilesChou

Outline

- PHP 8 更新簡介
- PHP 的未來

開始之前


祝 25 歲生日快樂

Java, Javascript, Delphi

注意：最終請參考 PHP 官網正式公告

PHP 8 更新五大重點

- 效能提升
- 語法擴充
- 彈性的強型別
- 更加嚴謹
- 調整內建套件與新增函式

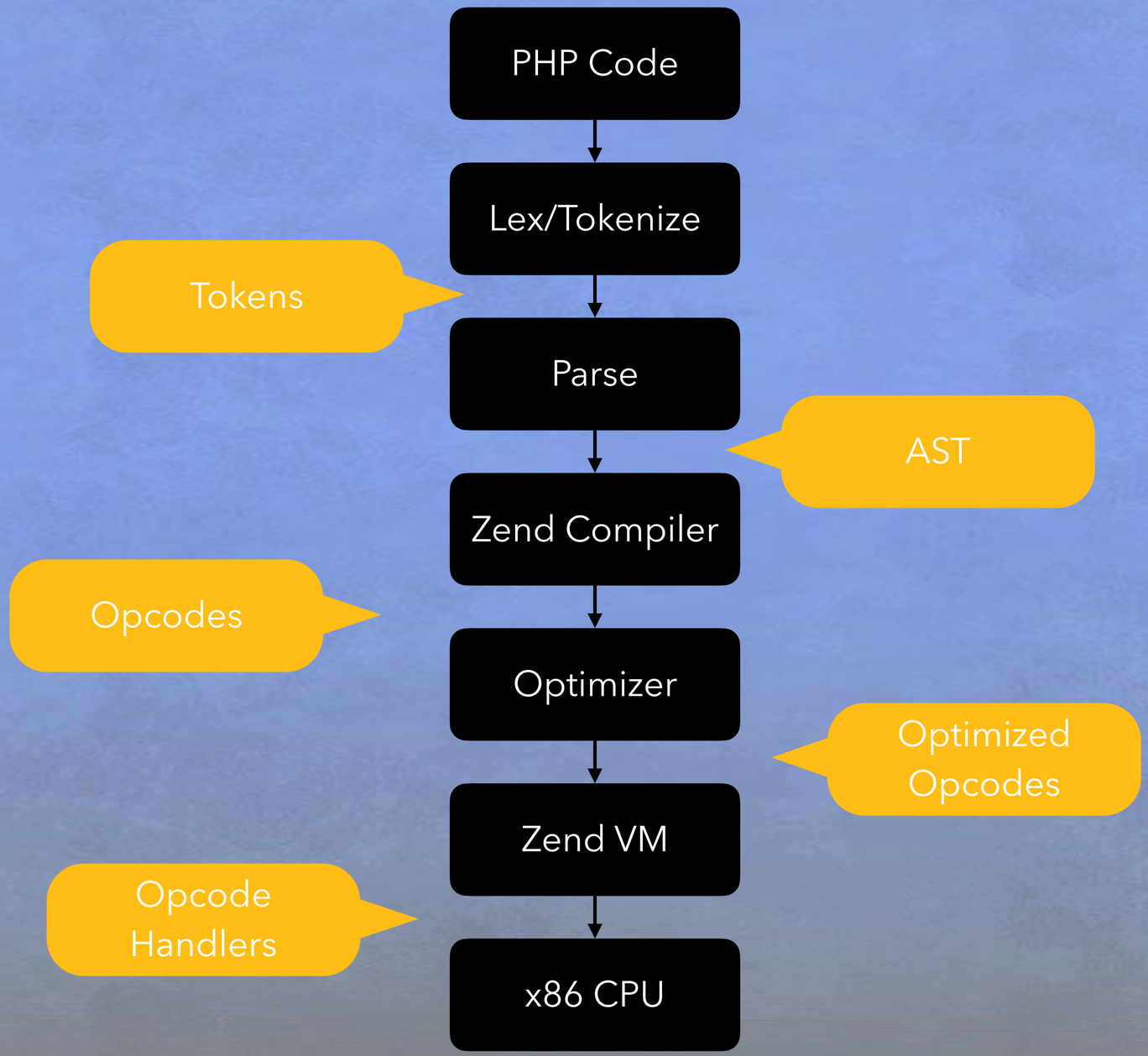
一、眾所期待的效能提升

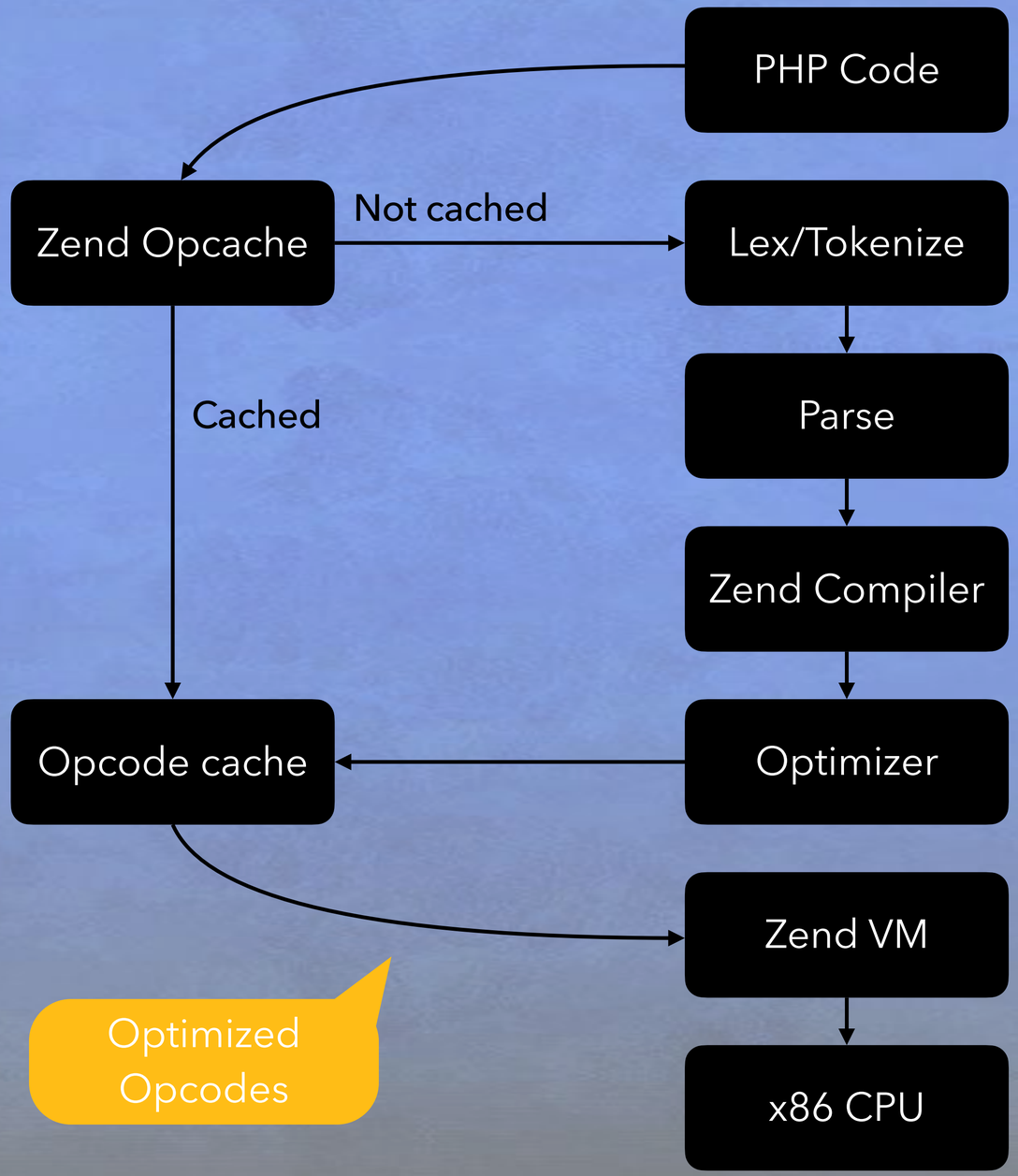
Just in Time Compiler

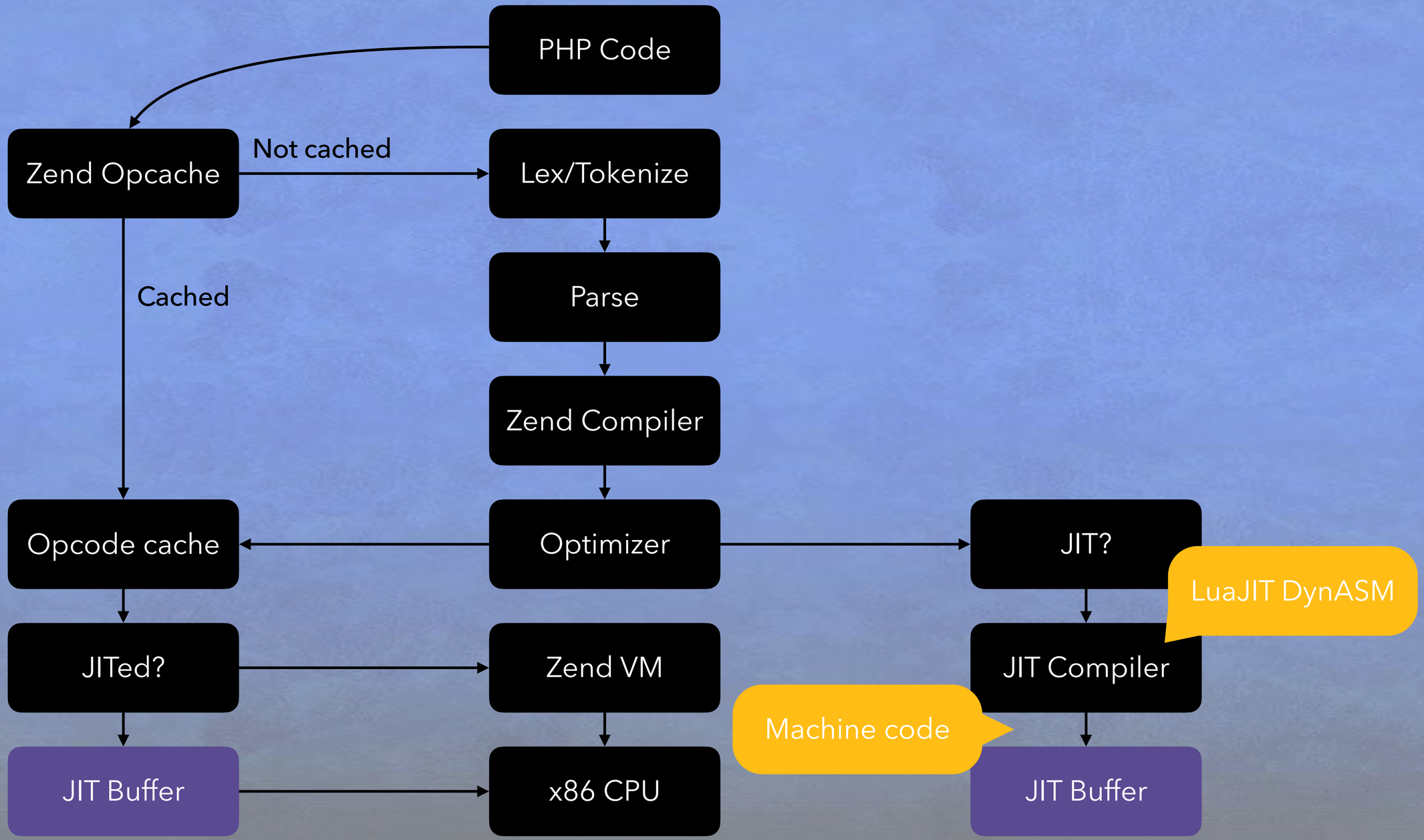
- 什麼是 JIT
- PHP 運作原理
- Opcache 運作原理
- JIT 運作原理
- JIT 小總結與建議

什麼是 JIT？

- 什麼是 AOT（Ahead-of-Time，事前編譯）
- 什麼是 interpreter（直譯器）
- 什麼是 JIT（Just-in-Time，動態編譯）


PHP Code
Lex/Tokenize
Tokens
Parse
AST
Zend Compiler
Opcodes
Optimizer
Optimized
Opcodes
Zend VM
Opcode
Handlers
x86 CPU


PHP Code
Zend Opcache
Not cached
Lex/Tokenize
Cached
Parse
Zend Compiler
Opcode cache
Optimizer
Zend VM
Optimized
Opcodes
x86 CPU


PHP Code
Zend Opcache
Not cached
Lex/Tokenize
Cached
Parse
Zend Compiler
Opcode cache
Optimizer
JIT?
LuaJIT DynASM
JITed?
Zend VM
JIT Compiler
Machine code
JIT Buffer
x86 CPU
JIT Buffer

Highlight 部分流程的行為

- Opcache 可以把編譯好的 Opcode 快取起來，留到下一次使用
- Preloading 可以在執行前把 PHP 編譯成 Opcode
- DynASM 可以把 Opcode 編譯成機器碼

為什麼不採用 AOT？

為什麼不採用 AOT？

- PHP 的 opcode 變數是弱型別，是由 Zend VM 做型別檢查
- 型別檢查編譯成機器碼執行，反而會更慢
- 把所有型別一次檢查完再全部編譯，則在編譯時會耗費 CPU 資源

opcache.jit 設定

- 使用四個數字如 1205，分別設定 CRTO 四個參數
  - CPU 是否啟用 AVX 指令集
  - 暫存器（Register）配置策略
  - 觸發（Trigger）時機
  - 最佳化（ Optimization）層級

注意 JIT 的限制

- 限 x86 CPU
- CPU-bound 才會有明顯效果
- 使用強型別寫法「理論上」會有明顯效果

DEMO

建議

- 多使用，並習慣強型別寫法，同時將有利於多人協作
- 先了解 CPU-bound 是什麼，與程式哪裡跟 CPU-bound 有關
- 決定要不要升級 PHP 8

Weak Maps

```php
class FooBar {
    private WeakMap $cache;

    public function getSomethingWithCaching(object $obj) {
        return $this->cache[$obj] ??= $this->computeSomethingExpensive($obj);
    }

    // ...
}
```

User Model

Decrypted Data

二、語法擴充

Match Expression

```php
switch ($thirdParty) {
    case 'Facebook':
        return redirect()->route( route: 'facebook.login');
    case 'GitHub':
        return redirect()->route( route: 'github.login');
    default:
        throw new UnknownThirdPartyException();
}
```

Match Expression

```php
return match ($thirdParty) {
    'Facebook' => redirect()->route('facebook.login'),
    'GitHub' => redirect()->route('github.login'),
    default => throw new UnknownThirdPartyException(),
}
```

Constructor Property Promotion

```php
private $x;

public function __construct(int $x)
{
    $this->x = $x;
}
```

```php
public function __construct(private int $x)
{
}
```

Attributes

```
use Opcache\Jit;


/**
 * @Jit
 */
function foo() {}
```

```
use Opcache\Jit;


<<Jit>>
function foo() {}
```

Allow ::class on objects

```php
$object = new stdClass;
var_dump($object::class); // "stdClass"
```

non-capturing catches

```
try {
    changeImportantData();
} catch (PermissionException) {
    echo "You don't have permission to do this";
}
```

三、型別有多強，由你決定

Mixed Type

```php
class A
{
    public function bar(): mixed {}
}
```

Union Types

```php
class Number {
    private int|float $number;

    public function setNumber(int|float $number): void {
        $this->number = $number;
    }

    public function getNumber(): int|float {
        return $this->number;
    }
}
```

Static return type

```php
class Test {
    public function doWhatever(): static {
        // Do whatever.
        return $this;
    }
}
```

Add Stringable interface

```php
interface Stringable
{
    public function __toString(): string;
}
```

四、明師出高徒，嚴謹出好 code

Consistent type errors for internal functions

```
75      protected function decrypt(Request $request)
76      {
77          foreach ($request->cookies as $key => $cookie) {
78              if ($this->isDisabled($key)) {
79                  continue;
80              }
81
82              try {
83                  $request->cookies->set($key, $this->decryptCooki
84              } catch (DecryptException $e) {
85                  $request->cookies->set($key, null);
86              }
87          }
88
89          return $request;
90      }
```

https://github.com/laravel/framework/blob/v7.21.0/src/Illuminate/Cookie/Middleware/EncryptCookies.php#L85

Consistent type errors for internal functions

```php
72      /**
73       * Sets an input by name.
74       *
75       * @param string|array $value
76       */
77      public function set(string $key, $value)
78      {
79          if (!is_scalar($value) && !\is_array($value) && !method_exists($value, '__toString')) {
80              trigger_deprecation('symfony/http-foundation', '5.1', 'Passing "%s" as a 2nd Argument to "%s()"
81          }
82
83          $this->parameters[$key] = $value;
84      }
```

https://github.com/symfony/symfony/blob/v5.1.2/src/Symfony/Component/HttpFoundation/InputBag.php#L79

Stricter type checks

```php
var_dump([] % [42]);
// int(0)
// WTF?
```

arithmetic/bitwise operators +, -, *, /, **, %, <<, >>, &, |, ^, ~, ++, --:

其他與嚴謹相關的 RFC

- Change Default PDO Error Mode
- Validation for abstract trait methods
- Pending Ensure correct signatures of magic methods

五、內建套件與函式庫調整

get_debug_type() vs. gettype()

```
Psy Shell v0.10.4 (PHP 8.0.0alpha2 — cli) by Justin Hileman
>>> gettype('str')
=> "string"
>>> get_debug_type('str')
=> "string"
>>>
>>> gettype(new stdClass())
=> "object"
>>> get_debug_type(new stdClass())
=> "stdClass"
>>>
```

新增 str_* 函式

- str_starts_with() vs. Str::startsWith()
- str_ends_with() vs. Str::endsWith()
- str_contains() vs. Str::contains()

Built-in extension / class 調整

- XML-RPC PHP 8 將移除，改放到 PECL 裡
- JSON PHP 7 可在編譯加參數安裝，到 PHP 8 將改為內建
- token_get_all() 改 PhpToken::getAll() 並讓回傳物件化

PHP 的未來

Pattern Match

```
+ var_dump(match ([1, 2, 3]) {
+      [false, $a] => wrong(),
+      [1, $a] => wrong(),
+      [1, 2, $a] => 'Literal pattern in array pattern: ' . $a,
+ });
```

Pattern Match

```
+ var_dump(match (15) {
+     0 ... 10 => wrong(),
+     10 ... 20 => 'Range pattern',
+     20 ... 30 => wrong(),
+ });
+
+ var_dump(match (true) {
+     false if false => wrong(),
+     false if true => wrong(),
+     true if false => wrong(),
+     true if true => 'Guard',
+ });
```

Pipe Operator

```php
$result = "Hello World" |> 'strlen';
```

```php
$result = strlen("Hello World");
```

Conditional Return

```php
if (empty($data)) {
    return [];
}

return $data;
```

```php
return if (empty($data)): [];

return $data;
```

Partial Function Application

```
function f(int $x, int $y): int {}

$partialFoo = f(?, 42);
```

is equivalent to

```
$partial = function(int $x): int {
    return f($x, 42);
};
```

總結

總結

- 新語法開始致敬其他 function programming 語言
- PHP 開始活用強型別寫法的特性
- PHP 正在一步一步的要求嚴謹
- JIT 目前在 PHP 8.0 能提升的情境有限


PHP
適合做的任務更多更廣了

WELCOME TO 104

JOIN US

DevOps Engineer

PHP Engineer