

# I Used to Hate PHP. Then I Got Good at It.

Or maybe PHP simply got better?

Sébastien Ballangé

Confoo - February 28th 2025



Feedback Form

# Who am I?

Staff Developer at DocuPet

~20 years of experience as a developer, mainly  
PHP and Symfony





# We're hiring!

<https://ca.indeed.com/cmp/Docupet-Inc.-3>

**What about you ?**



# 2004 PHP 5.0

or maybe 4.x ?

---

## 2004 - PHP 5.0

- Zend Engine 2
- better object model
  - visibility of properties and methods (no more \_ )
  - constructor and destructor
- exception handling
- SimpleXML
- MySQLi
- SQLite

## 2004 - PHP 4 <-> 5

```
class Foo
{
    var $_bar = 2;
    var $something;

    function Foo($bar, $something) {
        $this->_bar = $bar;
        $this->something = $something;
    }

    function _doNotUseThis() {
        // ...
    }

    function doSomething() {
        // ...
    }
}
```

```
class Foo
{
    private $bar = 2;
    public $something;

    public function __construct($bar, $something) {
        $this->bar = $bar;
        $this->something = $something;
    }

    private function doNotUseThis() {
        // ...
    }

    public function doSomething() {
        // ...
    }
}
```

## 2004 - PHP 5.0

- Who needs a router?
  - index.php
  - contact.php
  - ...
- Templates?
  - `include('header.php');`
- Magic quotes
- `register_globals`





## 2005 - PHP 5.1

- PDO
- improved dates management
- performance improvements

but also:

- CakePHP
- Symfony
- start of work on PHP 6 (Unicode)

## 2006 - PHP 5.2

- JSON
- DateTime classes
- ZIP
- input filters

but also:

- CodeIgniter
- Zend Framework

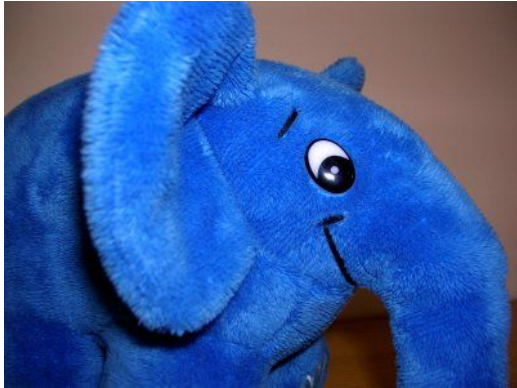
# 2007 - 2008



---

2007 - ... 🦗

- First elephant



# 2007 - ...

- Xdebug 2

<https://derickrethans.nl/xdebug-2-released.html>

## Comments

**Yann Larrivee**

*Wednesday, July 18th 2007, 19:52 UTC*

Congratulation Derick for this release. I will try this out shortly, when i return to our office.

By the way, i really like the new picture :)

See you in march! Yann

2008 - ... 

- Yii
- Doctrine

## 2007 - 2008, in my world

- Zend Server
- Sourceforge
- mix of custom autoloader custom and require\_once
- proprietary framework based on a C extension

# 2009 PHP 5.3



Modern PHP, kind of



## 2009 - PHP 5.3 - Elvis!

?:

```
$foo = $bar ?: "hello";
```

# 2009 - PHP 5.3 - namespaces

```
<?php
namespace MyProject\NamespaceA;

class MyClass {
    public function myMethod() {
        echo "This is MyClass in NamespaceA\n";
    }
}

namespace MyProject\NamespaceB;

use MyProject\NamespaceA\MyClass;

class AnotherClass {
    public function useClassFromNamespaceA() {
        $obj = new MyClass();
        $obj->myMethod();
    }
}

// Using a fully qualified name
\MyProject\NamespaceA\MyClass::myMethod();
```

# 2009 - PHP 5.3 - anonymous functions

```
class Foo {  
  
    private $message;  
  
    private function doSomething() {  
        $example = function () {  
            var_dump($this->message);  
        };  
        $example();  
    }  
}
```

=> Fatal error: Using \$this when not in object context

```
class Foo {  
  
    public $message;  
  
    public function doSomething() {  
        $that = $this;  
        $example = function () use ($that) {  
            var_dump($that->message);  
        };  
        $example();  
    }  
}
```

## 2009 - PHP 5.3 - goto

```
<?php  
  
goto a;  
echo 'Foo' ;  
  
a:  
echo 'Bar' ;  
  
?>  
  
=> Bar
```

## 2009 - PHP 5.3

- late static binding
- garbage collector

but also

- creation of PHP-FIG (PSR)

## meanwhile, in my world...

- RIP custom extension
- Ruby is the future!

# 2010 - 2011




---

## 2010 - ...

- PSR-0 (Autoloader)
- PhpStorm 1
- PHP 6 is abandoned
- Slim
- Doctrine 2
- FPM



2011 - ... 

- Composer 
- Symfony 2
- Laravel
- Guzzle
- Monolog

# 2011 - back to the future

- New job @ Manwin, in PHP 4 🤔
- New new job @ Manwin, in Symfony 😍
  - Dependency injection 🤖
  - Twig 😄
  - Doctrine 😐
  - Redis
  - Varnish

# **2012**

# **PHP 5.4**

---

## 2012 - PHP 5.4 - []

```
<?php
// Old syntax
$myArray = array('apple', 'banana', 'orange');

// Short array syntax (PHP 5.4+)
$myArray = ['apple', 'banana', 'orange'];

// Associative array with short syntax
$person = ['name' => 'Alice', 'age' => 30];
```

# 2012 - PHP 5.4 - traits

```
<?php
trait Logger {
    public function log($message) {
        echo "Logging: $message\n";
    }
}

class MyClass {
    use Logger;

    public function myMethod() {
        $this->log('This is a message from MyClass');
    }
}
```

```
<?php
$obj = new MyClass();
$obj->myMethod();

// => Logging: This is a message from MyClass
```

## 2012 - PHP 5.4

- `$this` in closures
- built-in server
- `register_globals`, magic quotes and safe mode are finally gone

but also:

- Zend Framework 2
- PSR-1 and PSR-2 (code style)

**2013**  
**PHP 5.5**

---

# 2013 - PHP 5.5 - generators

```
<?php

function fibonacciGenerator($limit) {
    $a = 0;
    $b = 1;

    for ($i = 0; $i < $limit; $i++) {
        yield $a;
        $c = $a + $b;
        $a = $b;
        $b = $c;
    }
}

foreach (fibonacciGenerator(10) as $num) {
    echo $num . ' ';
}

// 0 1 1 2 3 5 8 13 21 34
```



# 2013 - PHP 5.5 - finally

```
<?php

function divideNumbers($dividend, $divisor) {
    try {
        if ($divisor === 0) {
            throw new Exception("Division by zero");
        }

        return $dividend / $divisor;
    } catch (Exception $e) {
        echo "An error occurred: {$e->getMessage()}\n";
    } finally {
        echo "This block will always execute.\n";
    }
}
```

```
<?php

$result = divideNumbers(10, 2);
echo "Result: $result\n";
>
This block will always execute.
Result: 5

$result = divideNumbers(10, 0);
echo "Result: $result\n";
>
An error occurred: Division by zero
This block will always execute.
```

## 2013 - PHP 5.5 - misc

- `::class`
  - `\MyProject\MyClass::class`
  - but not `$myInstance::class` until PHP 8.0
- opcode caching with Zend OPcache
- `password_hash()` & `password_verify()`

# 2013 - new legacy

- New job @ DataCandy
  - Zend Framework 1 😭
  - No Composer
  - PHP 5.2
  - Automated tests (PHPUnit, SoapUI, ...)
  - Transactional system
  - Lots of APIs

# 2014 PHP 5.6

...

---

# 2014 - PHP 5.6 - variadic functions

```
<?php  
  
function sum(...$numbers) {  
    $total = 0;  
    foreach ($numbers as $number) {  
        $total += $number;  
    }  
    return $total;  
}  
  
$result = sum(1, 2, 3, 4, 5);  
echo $result; // Output: 15
```

# 2014 - PHP 5.6 - arguments unpacking

```
<?php  
function greet($greeting, $name, $surname) {  
    echo "$greeting, $name $surname!\n";  
}  
  
$details = ['Hi', 'Confoo', 'Attendees'];  
greet(...$details); // Output: Hi, Confoo Attendees!
```

## 2014 - PHP 5.6 - using \*\* for exponents

```
<?php  
  
$a = 5;  
  
$b = 2;  
  
$result = $a ** $b;  
  
echo $result; // 25
```

# 2014 - PHP 5.6

but also:

- PSR-4 replaces PSR-0
- Hack / HHVM by Facebook



# 2015 PHP 7.0

`declare(strict_types=1);`

---

## 2015 - PHP 7.0 - arguments and return types

```
<?php

class Calculator {
    public function add(int $a, int $b): int {
        return $a + $b;
    }
}

$calculator = new Calculator();
$result = $calculator->add(5, 3);
echo $result; // Output: 8
```

## 2015 - PHP 7.0 - anonymous classes

```
<?php

interface Logger {
    public function log(string $message);
}

$logger = new class implements Logger {
    public function log(string $message) {
        echo "Logging: $message\n";
    }
};

$logger->log("Hello, world!");
```

## 2015 - PHP 7.0 - spaceship operator <=>

```
<?php
$a = 10;
$b = 5;

$comparison = $a <=> $b;

if ($comparison > 0) {
    echo "$a is greater than $b";
} elseif ($comparison < 0) {
    echo "$a is less than $b";
} else {
    echo "$a is equal to $b";
}

// 10 is greater than 5
```

## 2015 - PHP 7.0 - null coalescing ??

```
<?php
$name = $inexistant ?? 'Guest';
echo "Hi, $name!";

// Hi, Guest!
```

# 2015 - PHP 7.0

but also:

- Error and Throwable
- `assert()`
- end for `mysql_*`()
- PHP 4 constructors deprecated

# 2016 PHP 7.1

---

## 2016 - PHP 7.1 - void

```
<?php

interface Logger {
    public function log(string $message): void;
}

$logger = new class implements Logger {
    public function log(string $message): void {
        echo "Logging: $message\n";
    }
};
```



## 2016 - PHP 7.1 - nullable type

```
<?php

class Logger {
    public function log(?string $message): void {
        $message = $message ?? "-";
        echo "Logging: $message\n";
    }
}

$logger->log(null); // Logging: -
```

# 2016 - PHP 7.1 - constants visibility

```
<?php

class MyClass {
    const PUBLIC_CONSTANT = 'This is a public constant';
    protected const PROTECTED_CONSTANT = 'This is a protected constant';
    private const PRIVATE_CONSTANT = 'This is a private constant';

    public function showConstants() {
        echo self::PUBLIC_CONSTANT . "\n";
        echo self::PROTECTED_CONSTANT . "\n";
        echo self::PRIVATE_CONSTANT . "\n";
    }
}

$obj = new MyClass();
echo $obj::PUBLIC_CONSTANT . "\n"; // Accessing public constant directly

// Accessing constants from within the class
$obj->showConstants();
```

# 2016 - PHP 7.1 - catch multiple exceptions

```
<?php

function divideNumbers($dividend, $divisor) {
    try {
        if ($divisor === 0) {
            throw new DivisionByZeroError("Division by zero error.");
        }

        return $dividend / $divisor;
    } catch (DivisionByZeroError | TypeError $e) {
        echo "An error occurred: " . $e->getMessage() . "\n";
    }
}



divideNumbers(10, 0); // An error occurred: Division by zero error.

divideNumbers(10, "hello"); // An error occurred: Unsupported operand types: int / string
```

# 2016 - PHP 7.1

- array unpacking using [] ( ⇔ list())

But also:

- Composer 1.0.0
- Zend Framework 3 
- PHPStan 

# 2017 PHP 7.2

Crypto stuff

---

## 2017 - PHP 7.2

- crypto: argon2 for password\_hash(), libsodium and removal of mcrypt
- type **object**
- end of HHVM for PHP

# 2017 - ZF1 won't die!

- Add Symfony components, one at a time
  - Console
  - DependencyInjection
  - Twig
  - Monolog
  - Doctrine
  - ...
- Foundation for a complete migration later
  - We'll do it after X is done...

**2018**  
**PHP 7.3**

---



# 2018 - PHP 7.3 - Heredoc and Nowdoc

```
<?php
$name = "Alice";
$message = <<<HEREDOC
    Hello, $name!
    This is a multi-line string
    using heredoc syntax.
    HEREDOC;

echo $message;

Hello, Alice!
This is a multi-line string using
heredoc syntax.
```

```
<?php
$message = <<<'NOWDOC'

This is a multi-line string using
nowdoc syntax.

Variable interpolation is not
allowed here.

NOWDOC;

echo $message;
```

## 2018 - PHP 7.3 - trailing commas in function calls

```
<?php  
  
divideNumbers(10, 2,);  
  
divideNumbers(  
    10,  
    2,  
);
```

# 2019 PHP 7.4

---

# 2019 - PHP 7.4 - typed properties

```
<?php

class Person {
    public string $name;
    public int $age;

    public function __construct(string $name, int $age) {
        $this->name = $name;
        $this->age = $age;
    }
}

$person = new Person("Alice", 30);
echo $person->name; // Output: Alice
echo $person->age; // Output: 30
```

# 2019 - PHP 7.4 - arrow functions

```
<?php
$numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5];
usort($numbers, fn($a, $b) => $a <=> $b);
print_r($numbers);
```

```
Array
(
    [0] => 1
    [1] => 1
    [2] => 2
    [3] => 3
    [4] => 3
    [5] => 4
    [6] => 5
    [7] => 5
    [8] => 5
    [9] => 6
    [10] => 9
)
```

## 2019 - PHP 7.4 - args unpacking in arrays

```
<?php
$parts = ['apple', 'pear'];
$fruits = ['banana', 'orange', ...$parts, 'watermelon'];
// ['banana', 'orange', 'apple', 'pear', 'watermelon'];
```

# 2019 - PHP 7.4 - Null coalescing assignment

```
<?php
$array['key'] ??= computeDefault();

// more or less equivalent to:

if (!isset($array['key'])) {
    $array['key'] = computeDefault();
}
```

# 2019 - PHP 7.4 - Numeric Literal Separator

```
<?php  
6.674_083e-11; // float  
299_792_458;   // decimal  
0xCAFE_F00D;   // hexadecimal  
0b0101_1111;   // binary
```



# 2019 - PHP 7.4

- Weak references
- preloading `opcache.preload=preload.php`
- `__serialize()` and `__unserialize()`

But also:

- Zend Framework => Laminas
- PSR-12

# 2020 PHP 8.0



## 2020 - PHP 8.0 - named arguments

```
<?php

myFunction(paramName: $value);

array_foobar(array: $value);

array_fill(0, count: 100, value: 50);
array_fill(value: 50, count: 100, start_index: 0);
```

## 2020 - PHP 8.0 - promoted properties

```
<?php

class Person {
    public function __construct(
        public string $name,
        private int $age,
    ) {}
}

$person = new Person("Alice", 30);
echo $person->name; // Output: Alice
echo $person->age; // Uncaught Error: Cannot access private property Person::$age
```

# 2020 - PHP 8.0 - attributes

```
<?php

#[Attribute]
class RequiresAuthentication {
    public function __construct(public bool $isAdmin = false) {}
}

class ProtectedPage {
    #[RequiresAuthentication(true)]
    public function adminOnly() {
        echo "This is an admin-only page.\n";
    }

    #[RequiresAuthentication]
    public function publicPage() {
        echo "This is a public page.\n";
    }
}
```

## 2020 - PHP 8.0 - union types

```
<?php

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

$calculator = new Calculator();
$result = $calculator->add(5, 3);
echo $result; // Output: 8

$result = $calculator->add(5, 2.5);
echo $result; // Output: 7.5
```

## 2020 - PHP 8.0 - match

```
<?php

$fruit = 'apple';

$result = match ($fruit) {
    'apple' => 'This is an apple.',
    'banana' => 'This is a banana.',
    'orange' => 'This is an orange.',
    default => 'Unknown fruit.'
};

echo $result; // Output: This is an apple.
```

## 2020 - PHP 8.0 - nullsafe operator ?->

```
<?php
```

```
$result = $repository?->getUser(5)?->name;
```



## 2020 - PHP 8.0 - Non-capturing catches

```
<?php

function divideNumbers($dividend, $divisor) {
    try {
        if ($divisor === 0) {
            throw new Exception("Division by zero");
        }

        return $dividend / $divisor;
    } catch (Exception) {
        echo "An error occurred.\n";
    }
}
```

# 2020 - PHP 8.0

But also:

- Just-In-Time compilation (JIT).
- static as a return type
- interface Stringable (if method `__toString()` exists)
- trailing commas in functions definition
- type mixed
- `str_contains()`, `str_starts_with()` and `str_ends_with()`
- class `Socket`
- PHP 4 constructors are gone

## 2020 - not much

- First time as a Confoo speaker
  - “From Legacy to Symfony”  
(more like “From Legacy to Legacy & Symfony”)
- Technical Lead @ DataCandy (now Paystone)
- New passion: adding features to a bot to annoy my coworkers

# **2021**

# **PHP 8.1**

---

## 2021 - PHP 8.1 - enums

```
<?php

enum Color {
    case Red;
    case Green;
    case Blue;
}

$favouriteColour = Color::Blue;

echo $favouriteColour->name; // Blue
```

```
<?php

enum Color: string {
    case Red = 'Rouge';
    case Green = 'Vert';
    case Blue = 'Bleu';
}

$favouriteColour = Color::Blue;

echo $favouriteColour->name; // Blue
echo $favouriteColour->value; // Bleu
```

# 2021 - PHP 8.1 - Fibers

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

<https://stitcher.io/blog/fibers-with-a-grain-of-salt>

# 2021 - PHP 8.1 - readonly properties

```
<?php

class Person {
    public function __construct(
        public readonly string $name,
        private int $age,
    ) {}
}

$person = new Person("Robert", 30);
echo $person->name; // Output: Robert
$person->name = 'Bob';
// Uncaught Error: Cannot modify readonly property Person::$name
```

# 2021 - PHP 8.1 - Pure intersection types (Foo&Bar)

```
<?php

interface LoggerInterface {
    public function log(string $message): void;
}

interface Configurable {
    public function configure(): void;
}

class AdvancedLogger implements LoggerInterface,
Configurable {
    // ... implementation ...
}
```

```
<?php

class MyService {
    public function processData(
        LoggerInterface&Configurable $logger,
    ): void {
        $logger->log("Processing data...");
        $logger->configure();
    }
}

$logger = new AdvancedLogger();
$service = new MyService();

$service->processData($logger);
```



## 2021 - PHP 8.1 - First-class callable syntax

```
<?php
$length = strlen(...);
echo $length('foo'); // 3

$callback = $this->myMethod(...);
$callback($request);
```

## 2021 - PHP 8.1

- never return type
- final class constants
- array\_is\_list()

# **2022**

# **PHP 8.2**

---

## 2021 - PHP 8.2 - readonly classes

```
<?php

readonly class Person {
    public function __construct(
        public string $name,
        private int $age,
    ) {}
}
```

## 2021 - PHP 8.2 - true, false and null types

```
<?php

interface Person {
    public function isAsleep(): bool;
    public function isAwake(): bool;
}

class Vampire implements Person {
    public function isAsleep(): false { return false; }
    public function isAwake(): true { return !$this->isAsleep(); }
}

$vampire = new Vampire();
var_dump($vampire->isAsleep(), $vampire->isAwake());

bool(false)
bool(true)
```

## 2022 - PHP 8.2

- `#[\SensitiveParameter]`
- constants in traits

# 2023 PHP 8.3



---

## 2023 - PHP 8.3 - typed class constants

```
<?php

class Config {
    public const string API_KEY = 'your_api_key';
    public const string BASE_URL =
'https://api.example.com';
    public const int MAX_RETRIES = 3;
}
```



## 2023 - PHP 8.3

- Anonymous classes can be readonly
- #[\Override]
- json\_validate()
- str\_increment() / str\_decrement()
- get\_class() / get\_parent\_class() without arguments are deprecated

# 2024 PHP 8.4

“What's new in PHP 8.4”  
by Derick Rethans

<https://confoo.ca/en/2025/session/what-s-new-in-php-8-4>

---

# 2024 - PHP 8.4

- Property hooks
- Asymmetric visibility
- **new** without parentheses
- Lazy objects
- **#[\Deprecated]**
- **array\_all()** / **array\_any()** / **array\_find()**
- HTML5 parser in **\Dom\HTMLDocument**
- Implicitly nullable parameter deprecated

# Thank You



Feedback Form

Slides: <https://github.com/confooca/2025>