



# Road to PHP 8

---

Mario Blazek aka Marek

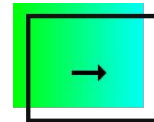
00

|

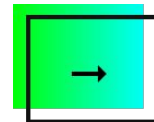
# Agenda

---

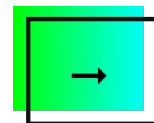
PHP 7.0



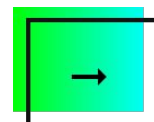
PHP 7.1



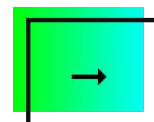
PHP 7.2



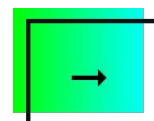
PHP 7.3



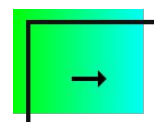
PHP 7.4



PHP 8



Demo time



01

|

# PHP 7.0

---

- **Scalar type declarations**
- **Return type declarations**
- **Null coalescing operator**
- **Spaceship operator**
- **Array constants**
- **Anonymous classes**

02

|

# Scalar type declarations

```
<?php
```

```
function sumOfInts(int ...$ints)
{
    return array_sum($ints);
}
```

- **string, int, float, bool**
- **PHP 5 - class names, interfaces, arrays, callable**

03

|

# Return type declarations

```
<?php
```

```
function sum(array ...$arrays): array
{
    return array_map(function(array $array): int {
        return array_sum($array);
    }, $arrays);
}
```

- **string, int, float, bool**
- **class names, interfaces, arrays, callable**

04

|

# Null coalescing operator

```
<?php
```

```
$username = $_GET['user'] ?? 'nobody';
```

```
$username = isset($_GET['user']) ? $_GET['user'] : 'nobody';
```

05

|

# Spaceship operator

```
<?php
// Integers
echo 1 <=> 1; // 0
echo 1 <=> 2; // -1
echo 2 <=> 1; // 1

// Floats
echo 1.5 <=> 1.5; // 0
echo 1.5 <=> 2.5; // -1
echo 2.5 <=> 1.5; // 1

// Strings
echo "a" <=> "a"; // 0
echo "a" <=> "b"; // -1
echo "b" <=> "a"; // 1
```

06

|

# Array constants

```
<?php
```

```
define('ANIMALS', [  
    'dog',  
    'cat',  
    'bird'  
]);
```

```
echo ANIMALS[1]; // outputs "cat"
```



07

|

# Anonymous classes

```
<?php
```

```
$app = new Application;  
$app->setLogger(new class implements Logger {  
    public function log(string $msg) {  
        echo $msg;  
    }  
});
```

# PHP 7.1

---

- **Nullable types**
- **Void functions**
- **Array destructuring with []**
- **Class constant visibility**
- **Iterable type**
- **Multi catch exception handling**

09

|

# Nullable types

```
<?php
```

```
function testReturn(): ?string
{
    return 'elePHPant';
}
```

```
function testReturn(): ?string
{
    return null;
}
```

10

|

# Void functions

```
<?php
```

```
function swap(&$left, &$right): void
{
    if ($left === $right) {
        return;
    }

    $tmp = $left;
    $left = $right;
    $right = $tmp;
}
```

11

|

# Array destructuring with []

```
<?php
```

```
$data = [  
    [1, 'Tom'],  
    [2, 'Fred'],  
];
```

```
list($id1, $name1) = $data[0];
```

```
[$id1, $name1] = $data[0];
```

```
foreach ($data as list($id, $name)) {  
    // logic here with $id and $name  
}
```

```
foreach ($data as [$id, $name]) {  
    // logic here with $id and $name  
}
```

12

|

# Class constant visibility

```
<?php
```

```
class ConstDemo
{
    const PUBLIC_CONST_A = 1;
    public const PUBLIC_CONST_B = 2;
    protected const PROTECTED_CONST = 3;
    private const PRIVATE_CONST = 4;
}
```

13

|

# Iterable pseudo-type

```
<?php
```

```
function iterator(iterable $iter)
{
    foreach ($iter as $val) {
        //
    }
}
```

- Object implementing Traversable
- array

14

|

# Multi catch exception handling

```
<?php
```

```
try {  
    // some code  
} catch (FirstException | SecondException $e) {  
    // handle first and second exceptions  
}
```



# PHP 7.2

---

- Object type
- Abstract method overriding
- Sodium as core extension
- Password hashing with Argon2
- Parameter type widening
- Trailing comma for grouped namespaces

16

|

# Object type

```
<?php
```

```
function test(object $obj) : object
{
    return new SplQueue();
}
```

```
test(new StdClass());
```

17

|

# Abstract method overriding

```
<?php
```

```
abstract class A
{
    abstract function test(string $s);
}
abstract class B extends A
{
    abstract function test($s) : int;
}
```

# **Sodium as core extension**

- **sodium\_\* functions**
- **SodiumException**

# Password hashing with Argon2

```
<?php
```

```
echo 'Argon2i hash: ' . password_hash('putinnukethem', PASSWORD_ARGON2I);
```

# Parameter type widening

```
<?php
```

```
interface A
{
    public function Test(array $input);
}
```

```
class B implements A
{
    public function Test($input){} // type omitted for $input
}
```

21

|

# Trailing comma for grouped namespaces

```
<?php
```

```
use Foo\Bar\{  
    Foo,  
    Bar,  
    Baz,  
};
```

# PHP 7.3

---

- Trailing commas in function calls
- The `is_countable` function
- Improved Heredoc syntax
- `array_key_first`
- `array_key_last`
- JSON errors can be thrown



23

|

# Trailing commas in function calls

```
<?php
```

```
unset (  
    $somethingToUnset,  
    $somethingElse,  
    $killIt,  
);
```

# Improved Heredoc

*// Instead of this:*

```
$query = <<<SQL
SELECT *
FROM `table`
WHERE `column` = true;
SQL;
```

*// You can do this:*

```
$query = <<<SQL
    SELECT *
    FROM `table`
    WHERE `column` = true;
SQL;
```

# JSON errors can be thrown

```
<?php
$json = 'invalid_json';

json_decode($json, JSON_THROW_ON_ERROR);
json_decode($json, false, 512, JSON_THROW_ON_ERROR);

// JsonException
```

# PHP 7.4

---

- Typed properties
- Arrow functions
- Limited return type covariance and argument type contravariance
- Null coalescing assignment operator
- Unpacking inside arrays
- Preloading

27

|

# Typed properties

```
<?php
```

```
class User {  
    public int $id;  
    public string $name;  
}
```

28

|

# Arrow functions

```
<?php
```

```
$y = 1;
```

```
$fn1 = fn($x) => $x + $y;
```

```
$fn2 = function ($x) use ($y) {  
    return $x + $y;  
};
```

# Limited return type covariance and argument type contravariance

```
<?php
```

```
class A {}
```

```
class B extends A {}
```

```
class Producer {
```

```
    public function method(): A {}
```

```
}
```

```
class ChildProducer extends Producer {
```

```
    public function method(): B {}
```

```
}
```

```
?>
```

30

|

# Null coalescing assignment operator

```
<?php
```

```
$array['key'] ??= computeDefault();  
// is roughly equivalent to  
if (!isset($array['key'])) {  
    $array['key'] = computeDefault();  
}
```



31

|

# Unpacking inside arrays

```
<?php
```

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

# Preloading

opcache.preload=our\_script.php

```
<?php
$directory = new RecursiveDirectoryIterator(__DIR__ . '/src');
$fullTree = new RecursiveIteratorIterator($directory);
$phpFiles = new RegexIterator($fullTree, '/.+((?<!Test)+\.php$)/i',
RecursiveRegexIterator::GET_MATCH);

foreach ($phpFiles as $key => $file) {
    require_once($file[0]);
}
```

- <https://symfony.com/blog/new-in-symfony-4-4-preloading-symfony-applications-in-php-7-4>

# PHP 8.0

---

- Union types
- The null safe operator
- Named arguments
- Attributes
- Match expression
- Constructor property promotion
- Mixed type
- Throw expression
- Weak maps
- ::class on objects
- Non capturing catches
- Trailing comma in parameter lists

# **Demo time...**

# Resources

- <https://github.com/MarioBlazek/road-to-php8>

**Thank you**