WORDCAMP
LISBOA 2023

From 5.3 to 8.2

What are you missing?

Lucas Giovanny

WORDCAMP
LISBOA 2023

Hi, I'm

Lucas Giovanny

@lucgiovanny

CTO @ vezoa

PHP Portugal Co-organizer

PHP SUPPORTED VERSIONS

VERSION	ACTIVE SUPPORT	SECUTIRY SUPPORT

VERSION	ACTIVE SUPPORT	SECUTIRY SUPPORT

PHP SUPPORTED VERSIONS

VERSION	ACTIVE SUPPORT	SECUTIRY SUPPORT
7.0	✗	✗
7.1	✗	✗
7.2	✗	✗
7.3	✗	✗
7.4	✗	✗

VERSION	ACTIVE SUPPORT	SECUTIRY SUPPORT

PHP SUPPORTED VERSIONS

VERSION	ACTIVE SUPPORT	SECUTIRY SUPPORT
7.0	✗	✗
7.1	✗	✗
7.2	✗	✗
7.3	✗	✗
7.4	✗	✗

VERSION	ACTIVE SUPPORT	SECUTIRY SUPPORT
8.0	✗	!

PHP SUPPORTED VERSIONS

VERSION	ACTIVE SUPPORT	SECUTIRY SUPPORT
7.0	✗	✗
7.1	✗	✗
7.2	✗	✗
7.3	✗	✗
7.4	✗	✗

VERSION	ACTIVE SUPPORT	SECUTIRY SUPPORT
8.0	✗	!
8.1	✓	✓

PHP SUPPORTED VERSIONS

VERSION	ACTIVE SUPPORT	SECUTIRY SUPPORT
7.0	✗	✗
7.1	✗	✗
7.2	✗	✗
7.3	✗	✗
7.4	✗	✗

VERSION	ACTIVE SUPPORT	SECUTIRY SUPPORT
8.0	✗	!
8.1	✓	✓
8.2	✓	✓

```php
<?php //7.4

class Pet {

    private string $name;

    public function setName(string $name)
    {
        $this->name = $name;
    }

    public function getName(): string
    {
        return $this->name;
    }
}
```

```php
<?php //7.4

class Pet {

    private string $name;

    public function setName(string $name)
    {
        $this->name = $name;
    }

    public function getName(): string
    {
        return $this->name;
    }
}
```

```php
<?php //7.4

class Pet {

    private string $name;

    public function setName(string $name)
    {
        $this->name = $name;
    }

    public function getName(): string
    {
        return $this->name;
    }
}
```

```php
<?php //7.4

class Pet {

    private string $name;

    public function setName(string $name)
    {
        $this->name = $name;
    }

    public function getName(): string
    {
        return $this->name;
    }
}

$dog = new Pet();
$dog->setName('Bingo');
var_dump($dog->getName()); // ‘Bingo' (string)
```

```php
<?php //7.4

class Game {

    /** @var int|float $score */
    private $score;

    /** @param int|float $score */
    public function setScore($score)
    {
        $this->score = $score;
    }

    /** @return int|float */
    public function getScore()
    {
        return $this->score;
    }
}
```

```php
<?php //7.4

class Game {

    /** @var int|float $score */
    private $score;

    /** @param int|float $score */
    public function setScore($score)
    {
        $this->score = $score;
    }

    /** @return int|float */
    public function getScore()
    {
        return $this->score;
    }
}

$game = new Game();
$game->setScore('100');
echo $game->getScore(); // '100' (string)
```

```php
<?php //7.4

class Game {

    /** @var int|float $score */
    private $score;

    /** @param int|float $score */
    public function setScore($score)
    {
        $this->score = $score;
    }

    /** @return int|float */
    public function getScore()
    {
        return $this->score;
    }
}
```

```php
<?php //7.4

class Game {

    /** @var int|float $score */
    private $score;

    /** @param int|float $score */
    public function setScore($score)
    {







        $this->score = $score;
    }

    /** @return int|float */
    public function getScore()
    {
        return $this->score;
    }
}
```

```php
<?php //7.4

class Game {

    /** @var int|float $score */
    private $score;

    /** @param int|float $score */
    public function setScore($score)
    {
        if (!is_int($score) && !is_float($score)) {
           throw new \InvalidArgumentException(
              sprintf('Argument $score should be either an integer or float, %s given', gettype($score))
           );
        }

        $this->score = $score;
    }

    /** @return int|float */
    public function getScore()
    {
        return $this->score;
    }
}
```

```php
<?php //7.4

class Game {

    /** @var int|float $score */
    private $score;

    /** @param int|float $score */
    public function setScore($score)
    {
        if (!is_int($score) && !is_float($score)) {
           throw new \InvalidArgumentException(
              sprintf('Argument $score should be either an integer or float, %s given', gettype($score))
           );
        }

        $this->score = $score;
    }

    /** @return int|float */
    public function getScore()
    {
        return $this->score;
    }
}

$game = new Game();

$game->setScore('100');

// InvalidArgumentException: Argument $score should be either an integer or float, string given
```

UNION TYPES

```php
<?php //8.0

class Game {

    private int|float $score;

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

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

```php
<?php //8.0

class Game {

    private int|float $score;

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

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

```php
<?php //8.0

class Game {

    private int|float $score;

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

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

```php
<?php //8.0

class Game {

    private int|float $score;

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

    public function getScore(): int|float
    {
        return $this->score;
    }
}

$game = new Game();
$game->setScore('100');

// Game::setScore(): Argument #1 ($score) must be of type int|float, string given
```

```php
<?php //8.0

class Collection {

    private array $items;

    public function addItem(array|bool|callable|int|float|null|object|string $value): void
    {
        $this->items[] = $value;
    }

    public function getLastItem(): array|bool|callable|int|float|null|object|string
    {
        return end($this->items);
    }
}
```

```php
<?php //8.0

class Collection {

    private array $items;

    public function addItem(array|bool|callable|int|float|null|object|string $value): void
    {
        $this->items[] = $value;
    }

    public function getLastItem(): array|bool|callable|int|float|null|object|string
    {
        return end($this->items);
    }
}
```

```php
<?php //8.0

class Collection {

    private array $items;

    public function addItem(array|bool|callable|int|float|null|object|string $value): void
    {
        $this->items[] = $value;
    }

    public function getLastItem():array|bool|callable|int|float|null|object|string
   {
        return end($this->items);
    }
}
```

```php
<?php //8.0

class Collection {

    private array $items;

    public function addItem(mixed $value): void
    {
        $this->items[] = $value;
    }

    public function getLastItem(): mixed
   {
        return end($this->items);
    }
}
```

WORDCAMP
LISBOA 2023

```php
<?php

$value = '2';

switch ($value) {
    case 0:
        $string = 'PHP';
        break;
    case 1:
        $string = ‘Word Camp';
        break;
    case 2:
        $string = ‘Lisboa’;
        break;
}
```

```php
<?php

$value = '2';

switch ($value) {
    case 0:
        $string = 'PHP';
        break;
    case 1:
        $string = ‘Word Camp';
        break;
    case 2:
        $string = ‘Lisboa’;
        break;
}


echo $string; // ‘Lisboa’
```

```php
<?php

$value = '2';

switch ($value) {
    case 0:
        $string = 'PHP';
        break;
    case 1:
        $string = ‘Word Camp';
        break;
    case 2:
        $string = ‘Lisboa’;
        break;
}


echo $string; // ‘Lisboa’
```

```php
<?php

$value = ‘2’;

switch ($value) {
    case 0:
        $string = 'PHP';
        break;
    case 1:
        $string = ‘Word Camp';
        break;
    case 2:
        $string = ‘Lisboa’;
        break;
}
```

```php
<?php

$value = ‘3’;

switch ($value) {
    case 0:
        $string = 'PHP';
        break;
    case 1:
        $string = ‘Word Camp';
        break;
    case 2:
        $string = ‘Lisboa’;
        break;
}
```

```php
<?php

$value = ‘3’;

switch ($value) {
    case 0:
        $string = 'PHP';
        break;
    case 1:
        $string = ‘Word Camp';
        break;
    case 2:
        $string = ‘Lisboa’;
        break;
}


echo $string; // Warning: Undefined variable $string
```

```php
<?php

$value = ‘3';

switch ($value) {
    case 0:
        $string = 'PHP';
        break;
    case 1:
        $string = ‘Word Camp';
        break;
    case 2:
        $string = ‘Lisboa';
        break;
```

```php
<?php

$value = ‘3';

switch ($value) {
    case 0:
        $string = 'PHP';
        break;
    case 1:
        $string = ‘Word Camp';
        break;
    case 2:
        $string = ‘Lisboa';
        break;
   default:
            throw new \InvalidArgumentException(sprintf('No `case` for $value %s', $value));
```

```php
<?php

$value = ‘3';

switch ($value) {
    case 0:
        $string = 'PHP';
        break;
    case 1:
        $string = ‘Word Camp';
        break;
    case 2:
        $string = ‘Lisboa';
        break;
   default:
            throw new \InvalidArgumentException(sprintf('No `case` for $value %s', $value));



   echo $string; // InvalidArgumentException: No `case` for $value 3
```

MATCH

```php
<?php

$value = 2;

echo match ($value) {
    0 ⇒ 'PHP',
    1 ⇒ ‘Word Camp',
    2 ⇒ ‘Lisboa’,
};
```

```php
<?php

$value = 2;

echo match ($value) {
    0 ⇒ 'PHP',
    1 ⇒ ‘Word Camp',
    2 ⇒ ‘Lisboa’,
}; // Lisboa
```

```php
<?php

$value = 2;

echo match ($value) {
    0 ⇒ 'PHP',
    1 ⇒ ‘Word Camp',
    2 ⇒ 'Lisboa',
}; // Lisboa
```

```php
<?php

$value = 2;

echo match ($value) {
    0 ⇒ 'PHP',
    1 ⇒ ‘Word Camp',
    2 ⇒ ‘Lisboa',
};
```

```php
<?php

$value = 3;

echo match ($value) {
    0 => 'PHP',
    1 => ‘Word Camp',
    2 => ‘Lisboa',
};

// UnhandledMatchError: Unhandled match value of type int
```

```php
<?php

$var = match ($value) {
    0 ⇒ 'PHP',
    1 ⇒ ‘Word Camp',
    2 ⇒ 'Lisboa',
};

function match($value)
{
  return match ($value) {
      0 ⇒ 'PHP',
      1 ⇒ ‘Word Camp',
      2 ⇒ ‘Lisboa',
  };
}
```

```php
<?php

$var = match ($value) {
    0 ⇒ 'PHP',
    1 ⇒ ‘Word Camp',
    2 ⇒ 'Lisboa',
};

function match($value)
{
  return match ($value) {
      0 ⇒ 'PHP',
      1 ⇒ ‘Word Camp',
      2 ⇒ ‘Lisboa’,
  };
}
```

```php
<?php

public function bar(int $age)
{
    return match (true) {
        $age <  18  => 'No beer',
        $age >= 18  => 'Yes, beer!',
    };
}

public function person(int $age)
{
    return match (true) {
        $age >= 65  => 'Senior',
        $age >= 25  => 'Adult',
        $age >= 18  => 'Young Adult',
        default     => 'Kid'
    };
}
```

```php
<?php

public function bar(int $age)
{
    return match (true) {
        $age <  18  => 'No beer',
        $age >= 18  => 'Yes, beer!',
    };
}

public function person(int $age)
{
    return match (true) {
        $age >= 65  => 'Senior',
        $age >= 25  => 'Adult',
        $age >= 18  => 'Young Adult',
        default     => 'Kid'
    };
}
```

```php
<?php

$x = 5;

$result = match ($x) {
    foo($x)          => true,
    $this->bar($x)   => false,
    1, 5             => baz(),
}
```

WORDCAMP
LISBOA 2023

```php
<?php

$items = [
    1, 2, 'Nuno', 5, 'Caneco'
];

$filter = array_filter($items, fn (mixed $item): bool => is_int($item));
```

```php
<?php

$items = [
    1, 2, 'Nuno', 5, 'Caneco'
];

$filter = array_filter($items, fn (mixed $item): bool => is_int($item));
```

```php
<?php

$items = [
    1, 2, 'Nuno', 5, 'Caneco'
];

$filter = array_filter($items, fn (mixed $item): bool => is_int($item));
```

```php
<?php

$items = [
    1, 2, 'Nuno', 5, 'Caneco'
];

$filter = array_filter($items, fn (mixed $item): bool => is_int($item));
```

```php
<?php

$items = [
    1, 2, 'Nuno', 5, 'Caneco'
];

$filter = array_filter($items, fn (mixed $item): bool => is_int($item));


var_dump($filter);
// [1, 2, 5]
```

```php
<?php

var_dump(
    array_fill(2, 3, 'Free Pizza 🍕')
);
```

```php
<?php

var_dump(
    array_fill(2, 3, 'Free Pizza 🍕')
);

// [2 ⇒ 'Free Pizza 🍕', 3 ⇒ 'Free Pizza 🍕', 4 ⇒ 'Free Pizza 🍕']
```

NAMED ARGUMENTS

```php
<?php

var_dump(
    array_fill(2, 3, 'Free Pizza 🍕')
);

// [2 ⇒ 'Free Pizza 🍕', 3 ⇒ 'Free Pizza 🍕', 4 ⇒ 'Free Pizza 🍕']
```

```php
<?php

var_dump(
    array_fill(2, 3, 'Free Pizza 🍕')
);

// [2 => 'Free Pizza 🍕', 3 => 'Free Pizza 🍕', 4 => 'Free Pizza 🍕']

var_dump(
    array_fill(start_key: 2, count: 3, value: 'Free Pizza 🍕')
);
```

```php
<?php

class Employee
{
    public function __construct(string $name, string $sector)
    {
        // ...
    }
}
```

```php
<?php

class Employee
{
    public function __construct(string $name, string $sector)
    {
        // ...
    }
}

new Employee(name: 'Lucas Giovanny', sector: 'IT');
```

```php
<?php

class User
{
    public function __construct(
        bool $admin,
        bool $active,
        ?array $rules = [],
    ) {
        // ...
    }
```

```php
<?php

class User
{
    public function __construct(
        bool $admin,
        bool $active,
        ?array $rules = [],
    ) {
        // ...
    }


$user = new User(true, false, null);
```

```php
<?php

class User
{
    public function __construct(
        bool $admin,
        bool $active,
        ?array $rules = [],
    ) {
        // ...
    }
}

$user = new User(true, false, null);
```

```php
<?php

class User
{
    public function __construct(
        bool $admin,
        bool $active,
        ?array $rules = [],
    ) {
        // ...
    }
}

$user = new User(true, false, null);

$user = new User(
    admin: true,
    active: false,
    rules: null
);
```

```php
<?php

function randomOrder(int $a, string $b, float $c)
{
    // ...
}
```

```php
<?php

function randomOrder(int $a, string $b, float $c)
{
    // ...
}

randomOrder(b: 'PHP', c: 8.0, a: 1);
```

```php
<?php

function optionals(string $a = 'default', int $b = 1)
{
    // ...
}
```

```php
<?php

function optionals(string $a = 'default', int $b = 1)
{
    // ...
}


optionals(b: 3);
```

```php
<?php


function skip(bool $production, string $language, float $version)
{
    // ...
}
```

```php
<?php


function skip(bool $production, string $language, float $version)
{
    // ...
}

skip(1, 'PHP', version: 8.0);
```

```php
<?php


function skip(bool $production, string $language, float $version)
{
    // ...
}

skip(1, 'PHP', version: 8.0);

skip(1, language: 'PHP', 8.0);

// Fatal error: Cannot use positional argument after named argument
```

WORDCAMP
LISBOA 2023

```php
<?php

class Status {

    protected array $status = [
        0 => 'DRAFT',
        1 => 'PUBLISHED',
        2 => 'ARCHIVED'
    ];

    public static function status(int|string $status): int
    {
        if(is_int($status)){
            return self::status[$status] ?? throw new InvalidArgumentException(
                sprintf('No status id %s on possible status', $status)
            );
        }

        return array_search(needle: $status, array: self::status) ?? throw new InvalidArgumentException(
            sprintf('No status value %s on possible status', $status)
        );
    }
}

$blogPost = new BlogPost(Status::status('DRAFT'));
```

```php
<?php

class Status {

    protected array $status = [
        0 => 'DRAFT',
        1 => 'PUBLISHED',
        2 => 'ARCHIVED'
    ];

    public static function status(int|string $status): int
    {
        if(is_int($status)){
            return self::status[$status] ?? throw new InvalidArgumentException(
                sprintf('No status id %s on possible status', $status)
            );
        }

        return array_search(needle: $status, array: self::status) ?? throw new InvalidArgumentException(
            sprintf('No status value %s on possible status', $status)
        );
    }
}

$blogPost = new BlogPost(Status::status(‘DRAFT'));
```

```php
<?php

class Status {

    protected array $status = [
        0 => 'DRAFT',
        1 => 'PUBLISHED',
        2 => 'ARCHIVED'
    ];

    public static function status(int|string $status): int
    {
        if(is_int($status)){
            return self::status[$status] ?? throw new InvalidArgumentException(
                sprintf('No status id %s on possible status', $status)
            );
        }

        return array_search(needle: $status, array: self::status) ?? throw new InvalidArgumentException(
            sprintf('No status value %s on possible status', $status)
        );
    }
}

$blogPost = new BlogPost(Status::status('DRAFT'));
```

```php
<?php

class Status {

    protected array $status = [
        0 => 'DRAFT',
        1 => 'PUBLISHED',
        2 => 'ARCHIVED'
    ];

    public static function status(int|string $status): int
    {
        if(is_int($status)){
            return self::status[$status] ?? throw new InvalidArgumentException(
                   sprintf('No status id %s on possible status', $status)
               );
        }

        return array_search(needle: $status, array: self::status) ?? throw new InvalidArgumentException(
            sprintf('No status value %s on possible status', $status)
        );
    }
}

$blogPost = new BlogPost(Status::status('DRAFT'));
```

```php
<?php

class Status {

    protected array $status = [
        0 => 'DRAFT',
        1 => 'PUBLISHED',
        2 => 'ARCHIVED'
    ];

    public static function status(int|string $status): int
    {
        if(is_int($status)){
            return self::status[$status] ?? throw new InvalidArgumentException(
                sprintf('No status id %s on possible status', $status)
            );
        }

        return array_search(needle: $status, array: self::status) ?? throw new InvalidArgumentException(
            sprintf('No status value %s on possible status', $status)
        );
    }
}

$blogPost = new BlogPost(Status::status('DRAFT'));
```

```php
<?php

class Status {

    protected array $status = [
        0 => 'DRAFT',
        1 => 'PUBLISHED',
        2 => 'ARCHIVED'
    ];

    public static function status(int|string $status): int
    {
        if(is_int($status)){
            return self::status[$status] ?? throw new InvalidArgumentException(
                sprintf('No status id %s on possible status', $status)
            );
        }

        return array_search(needle: $status, array: self::status) ?? throw new InvalidArgumentException(
            sprintf('No status value %s on possible status', $status)
        );
    }
}

$blogPost = new BlogPost(Status::status('DRAFT'));
```

ENUM

```php
<?php

enum Status
{
    case DRAFT = 0;
    case PUBLISHED = 1;
    case ARCHIVED = 2;
}
```

```php
<?php

enum Status
{
    case DRAFT = 0;
    case PUBLISHED = 1;
    case ARCHIVED = 2;
}


$value = Status::PUBLISHED->value; // ‘2'
```

```php
<?php

enum Status
{
    case DRAFT = 0;
    case PUBLISHED = 1;
    case ARCHIVED = 2;
}


$value = Status::PUBLISHED->value;  // '2'

$name  = Status::from(0)->name; // 'DRAFT'
```

```php
<?php

enum Status
{
    case DRAFT = 0;
    case PUBLISHED = 1;
    case ARCHIVED = 2;
}


$value = Status::PUBLISHED->value; // ‘2'

$name  = Status::from(0)->name; // ‘DRAFT’

class BlogPost
{
    public function __construct(
        public Status $status,
    ) {
        // ...
    }
}
```

```php
<?php

enum Status
{
    case DRAFT = 0;
    case PUBLISHED = 1;
    case ARCHIVED = 2;
}


$value = Status::PUBLISHED->value; // ‘2'

$name  = Status::from(0)->name; // ‘DRAFT’

class BlogPost
{
    public function __construct(
        public Status $status,
    ) {
        // ...
    }
}

$post = new BlogPost(Status::DRAFT);
```

```php
<?php

enum Status
{
    case DRAFT = 0;
    case PUBLISHED = 1;
    case ARCHIVED = 2;









}
```

```php
<?php

enum Status
{
    case DRAFT = 0;
    case PUBLISHED = 1;
    case ARCHIVED = 2;

    public function color(): string
        {
            return match($this)
            {
                Status::DRAFT => 'grey',
                Status::PUBLISHED => 'green',
                Status::ARCHIVED => 'red',
            };
        }

}
```

```php
<?php

enum Status
{
    case DRAFT = 0;
    case PUBLISHED = 1;
    case ARCHIVED = 2;

    public function color(): string
        {
            return match($this)
            {
                Status::DRAFT => 'grey',
                Status::PUBLISHED => 'green',
                Status::ARCHIVED => 'red',
            };
        }

}

$status = Status::ARCHIVED;

$status->color(); // 'red'
```

Thanks!
Any questions?

Lucas Giovanny

Vezoa, CTO
@lucgiovanny

lucas@lucasgiovanny.com