PHPStan sur du legacy
Un puits sans fond ?

Par Frédéric Bouchery de CCMBenchmark

@FredBouchery
(sur X, Bluesky ou Mastodon)

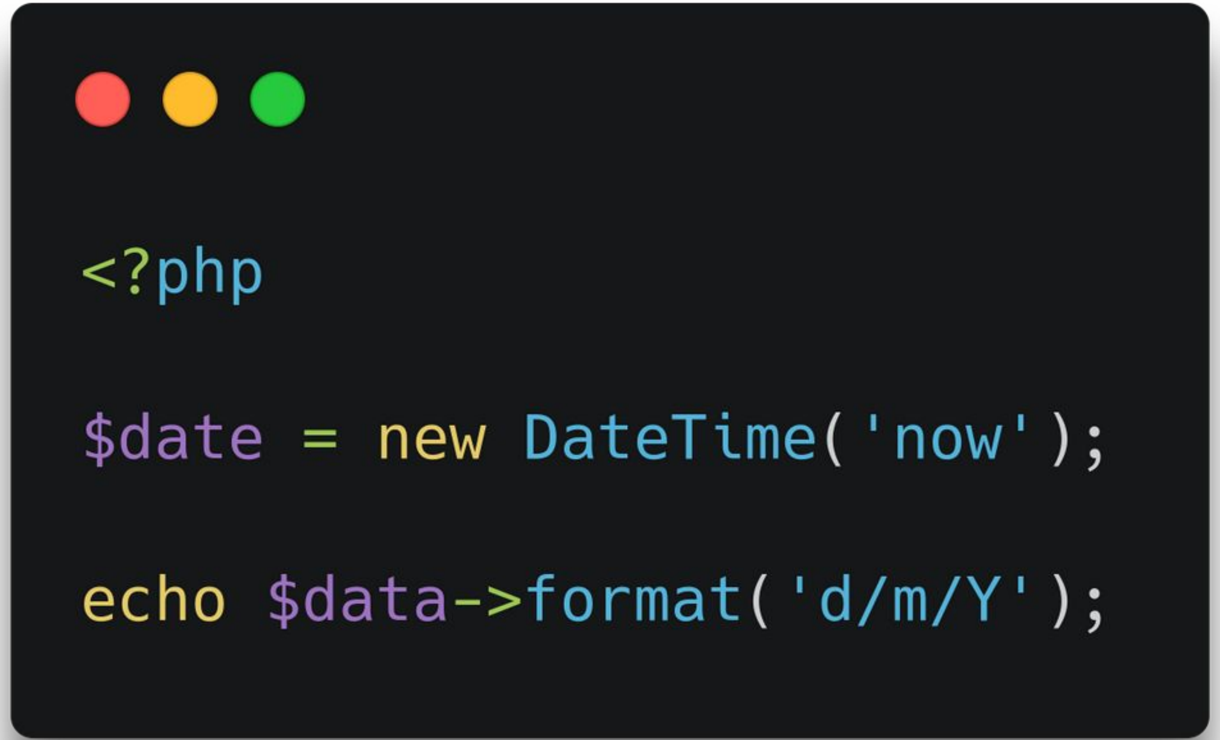
DRUPALCAMP
ROAZHON
Rennes - 28 au 30 mars 2024

your high-digital place open AXESS Digital & Business Experts SMILE Talan

Pourquoi faire de l’analyse statique ?

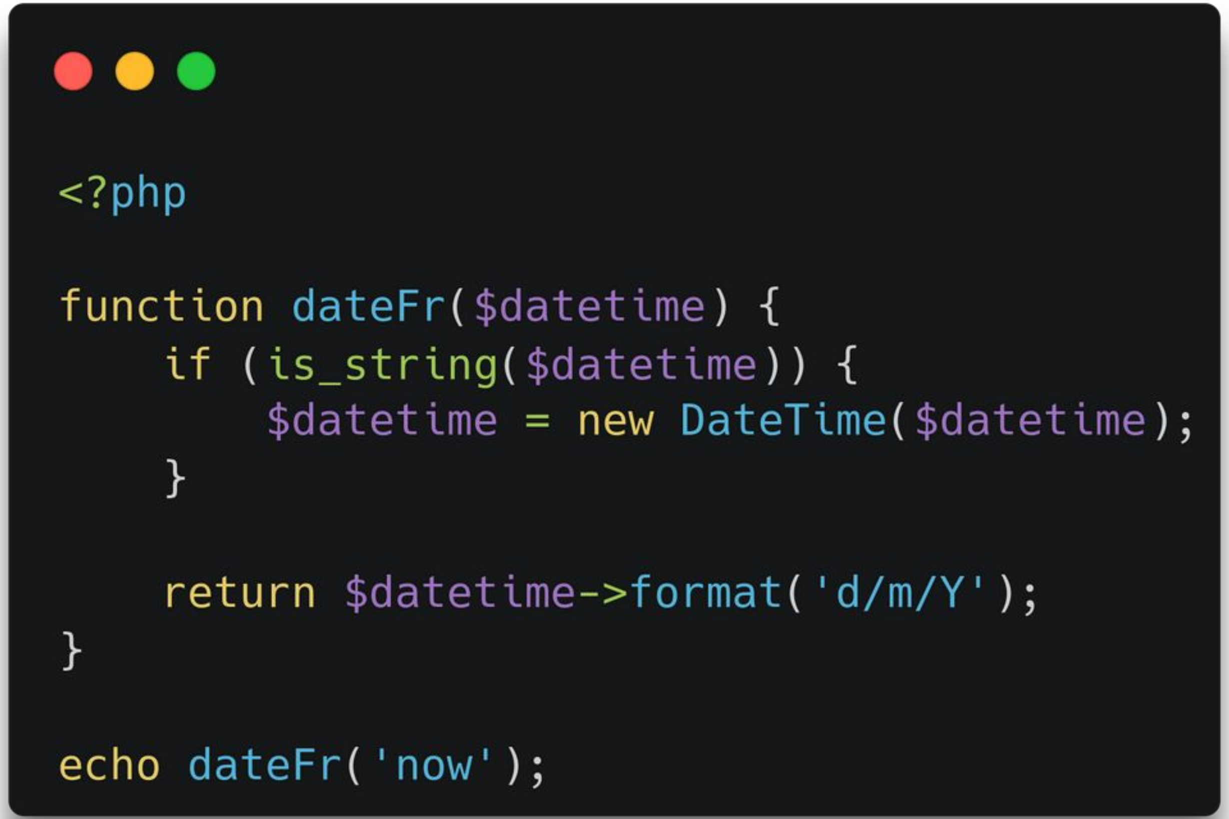
Pourquoi ?

Pourquoi faire de l’analyse statique ?



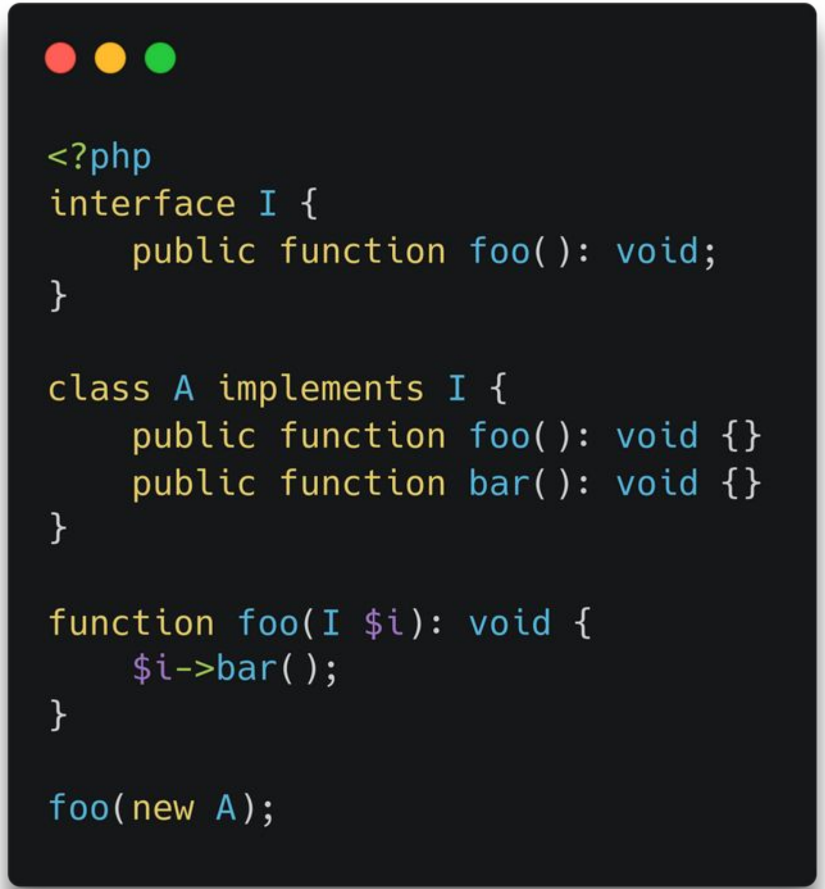
```php
<?php

$date = new DateTime('now');

echo $data->format('d/m/Y');
```

Pourquoi faire de l’analyse statique ?



```php
<?php

function dateFr($datetime) {
    if (is_string($datetime)) {
        $datetime = new DateTime($datetime);
    }

    return $datetime->format('d/m/Y');
}

echo dateFr('now');
```

Pourquoi faire de l’analyse statique ?



```php
<?php
interface I {
    public function foo(): void;
}

class A implements I {
    public function foo(): void {}
    public function bar(): void {}
}

function foo(I $i): void {
    $i->bar();
}

foo(new A);
```

PHPStan

Psalm

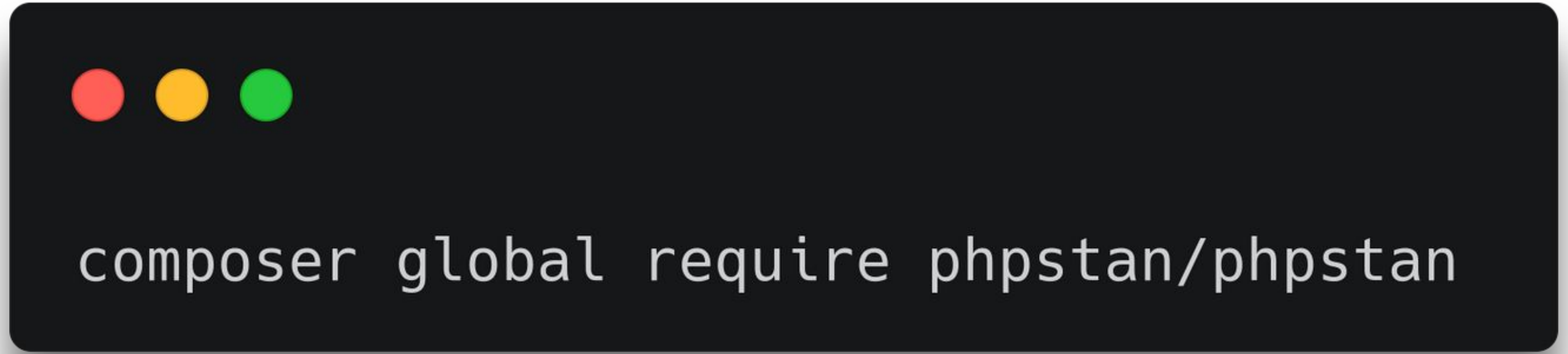
Scrutinizer

Phan

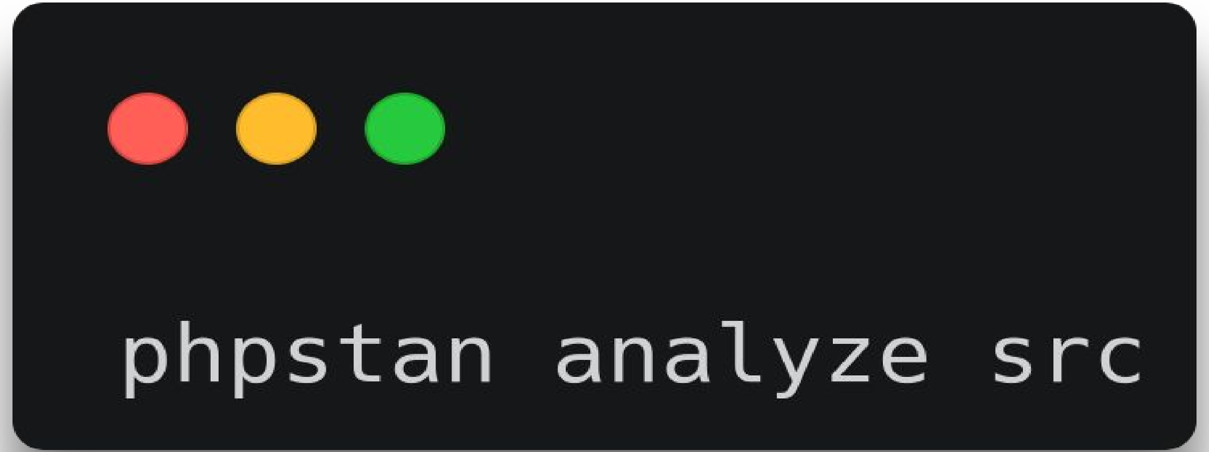
PHP Mess Detector

Sonarqube

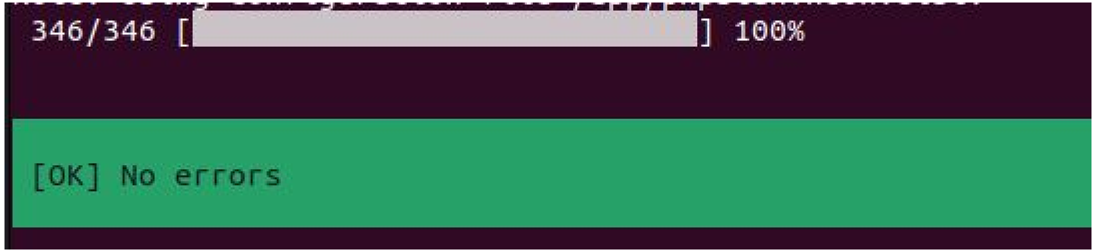
PhpCsFixer



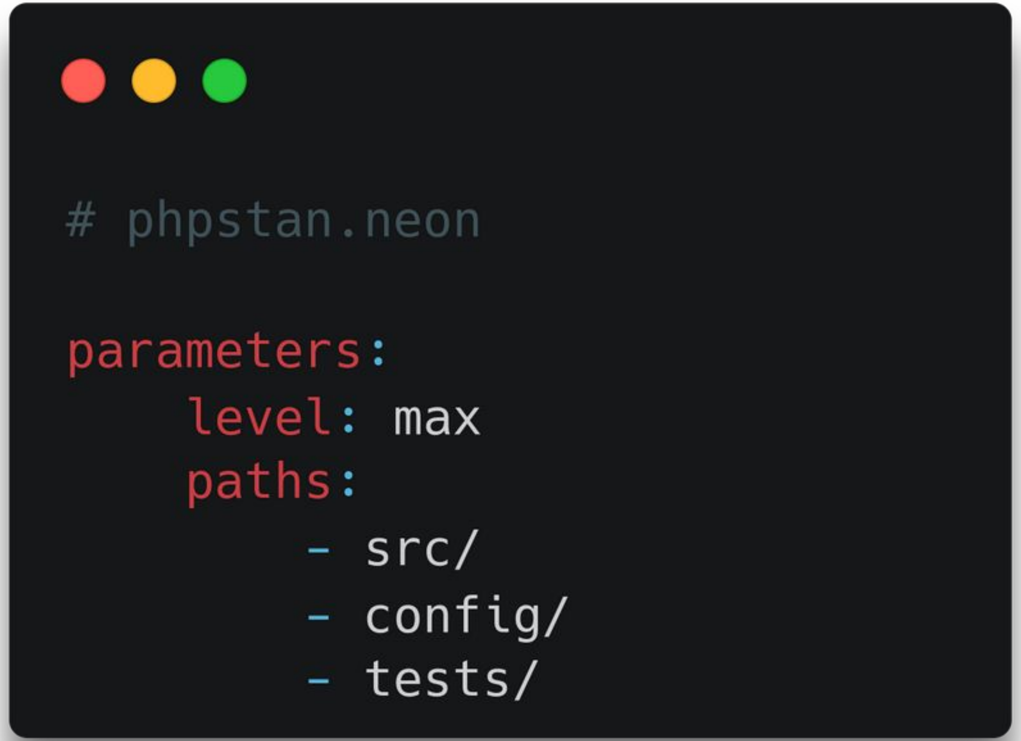
```
composer global require phpstan/phpstan
```



```
phpstan analyze src
```



```
 346/346 [▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓] 100%


 [OK] No errors
```

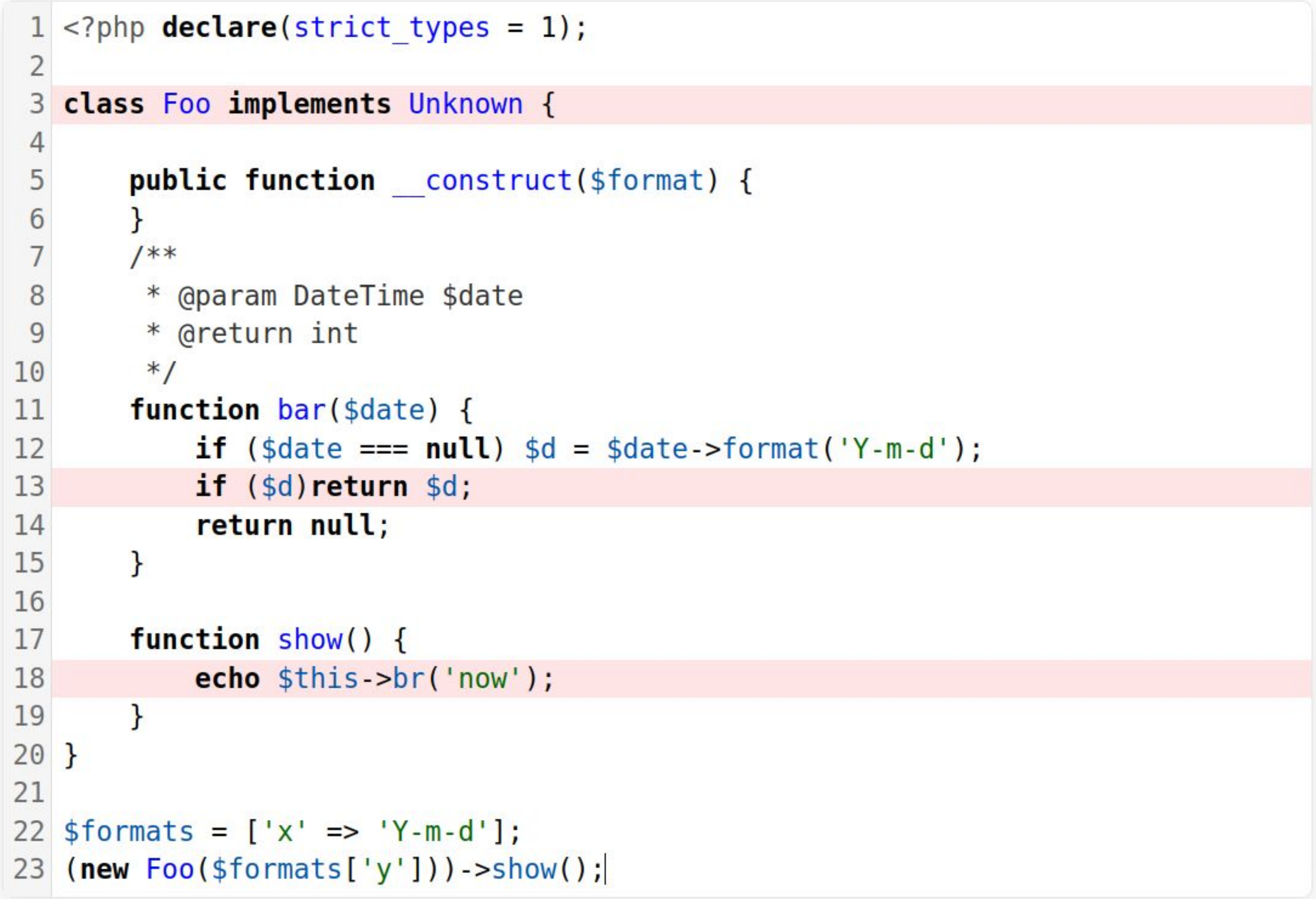


```
# phpstan.neon

parameters:
    level: max
    paths:
        - src/
        - config/
        - tests/
```

Quel niveau d’erreur ?

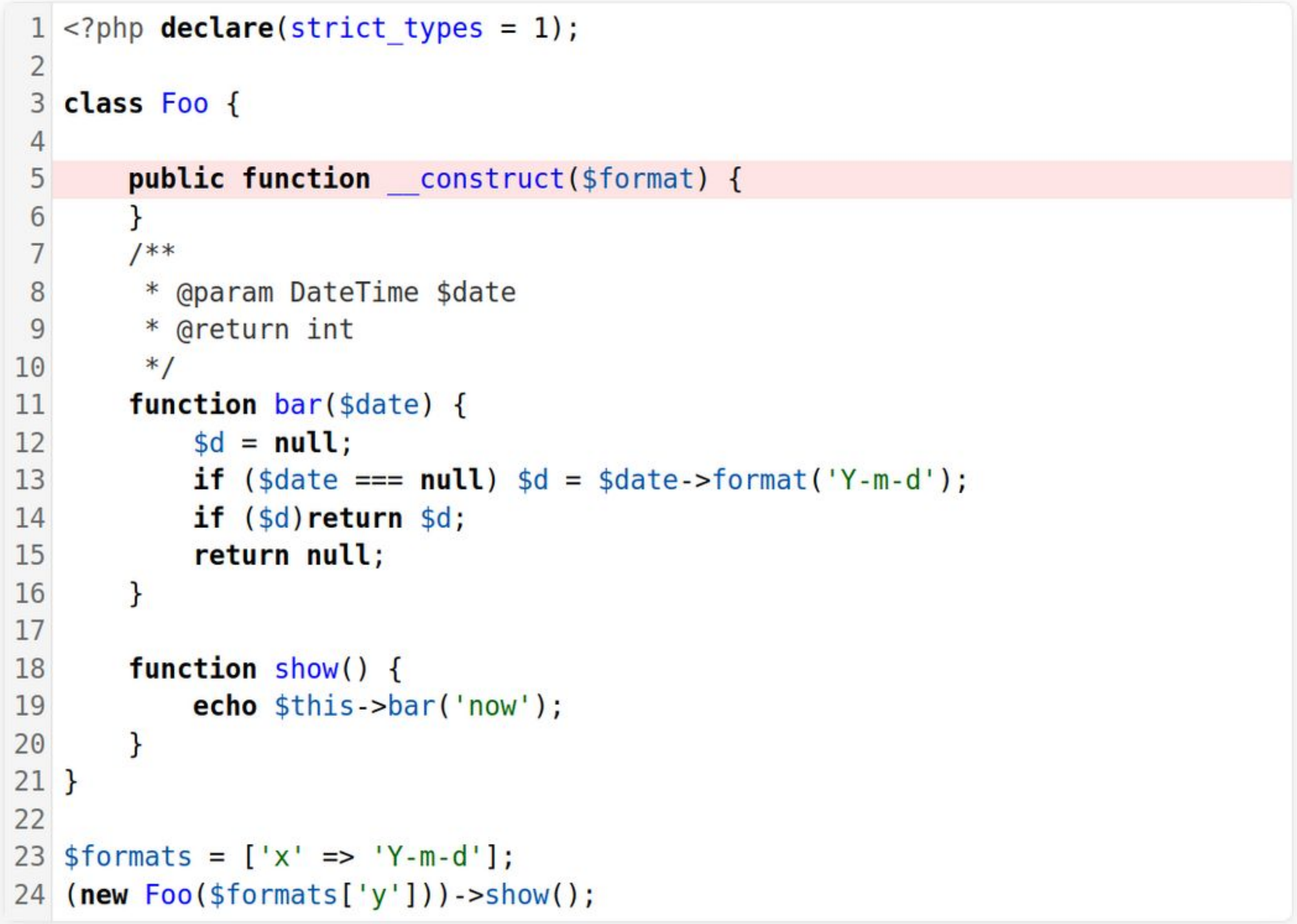
De la plus grossière à la plus subtile
(ou pas)

Level:
0



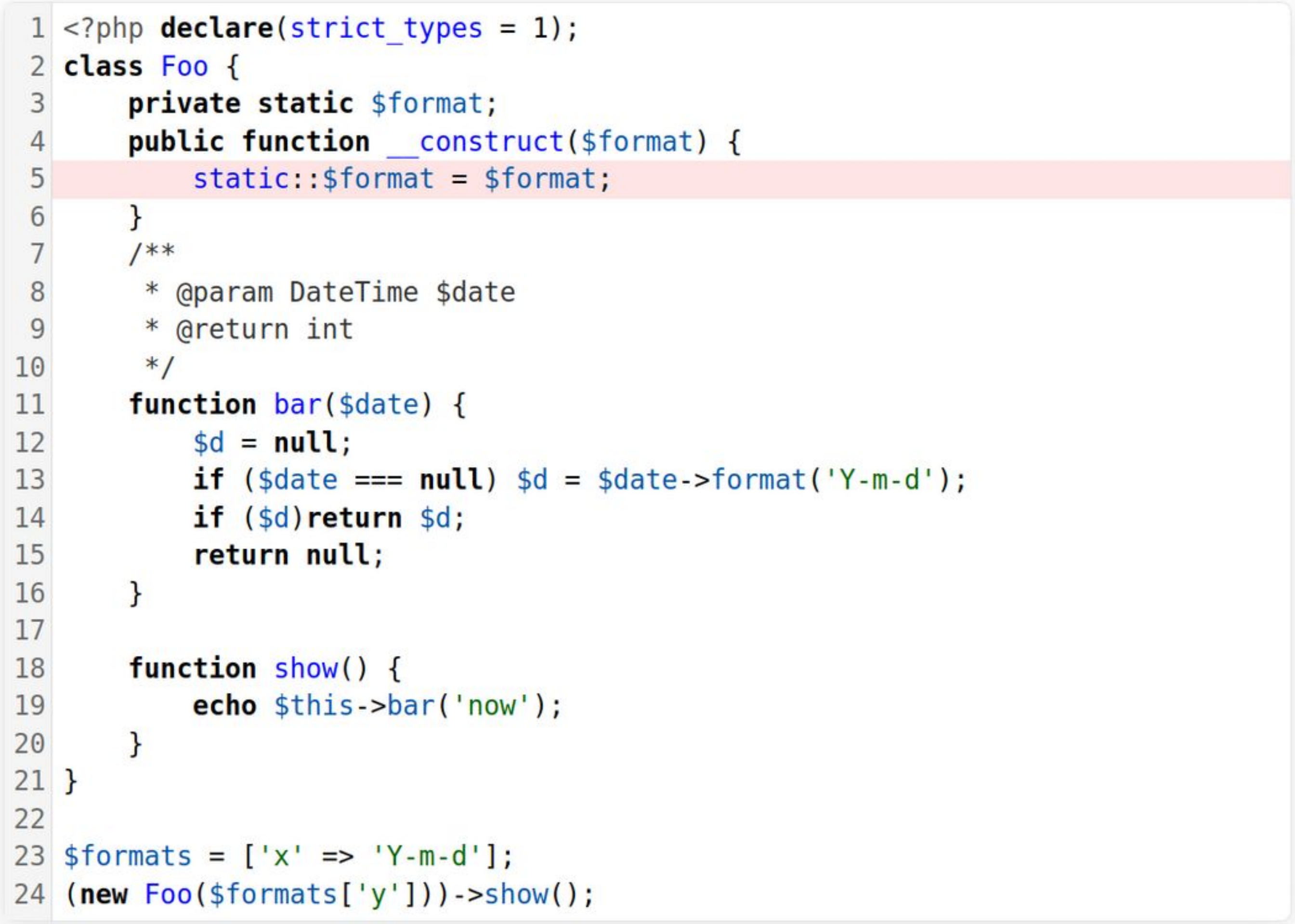
```php
<?php declare(strict_types = 1);

class Foo implements Unknown {

    public function __construct($format) {
    }
    /**
     * @param DateTime $date
     * @return int
     */
    function bar($date) {
        if ($date === null) $d = $date->format('Y-m-d');
        if ($d)return $d;
        return null;
    }

    function show() {
        echo $this->br('now');
    }
}

$formats = ['x' => 'Y-m-d'];
(new Foo($formats['y']))->show();
```

Level: 1



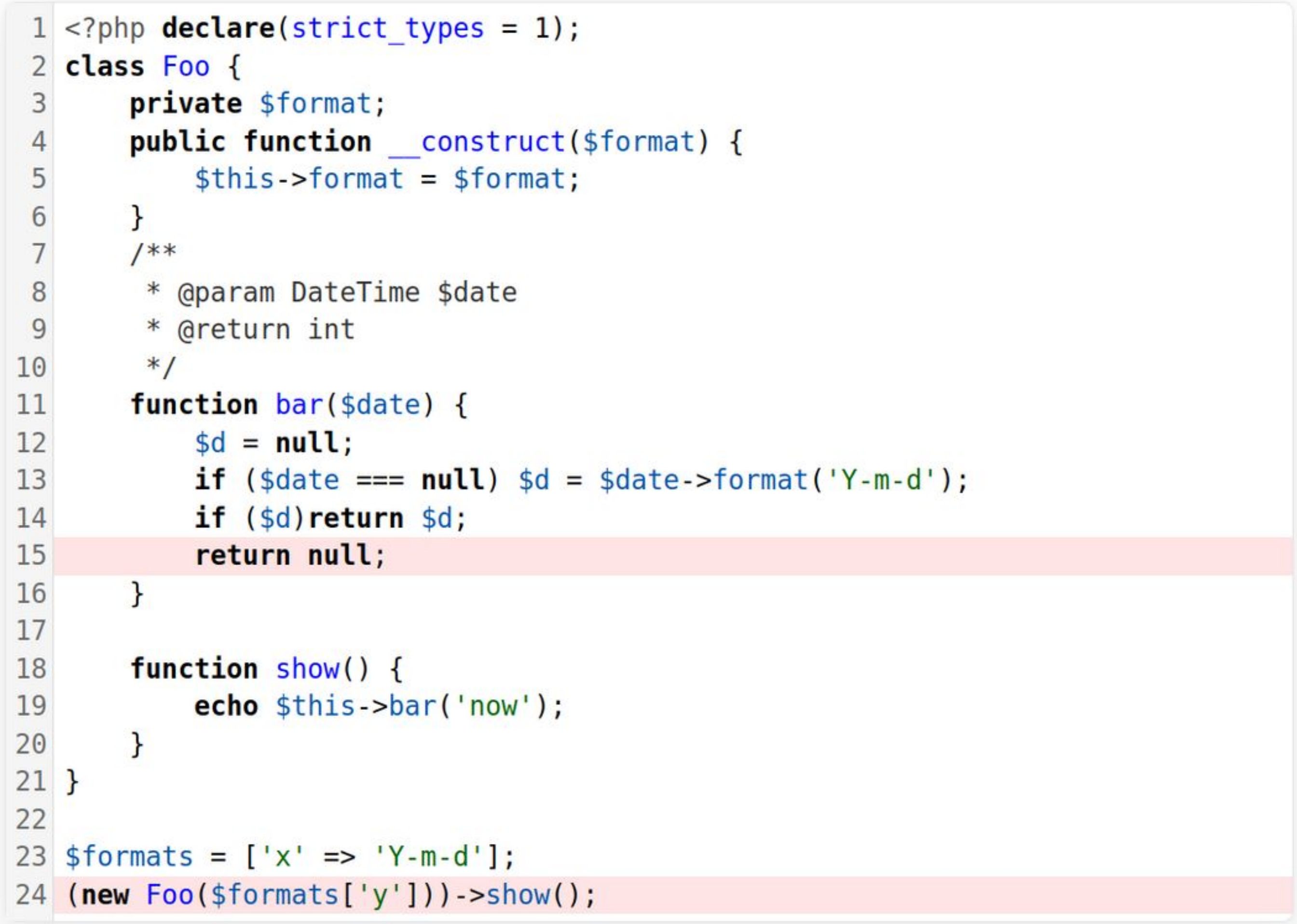
```php
<?php declare(strict_types = 1);

class Foo {

    public function __construct($format) {
    }
    /**
     * @param DateTime $date
     * @return int
     */
    function bar($date) {
        $d = null;
        if ($date === null) $d = $date->format('Y-m-d');
        if ($d)return $d;
        return null;
    }

    function show() {
        echo $this->bar('now');
    }
}

$formats = ['x' => 'Y-m-d'];
(new Foo($formats['y']))->show();
```

Level: 2



```php
<?php declare(strict_types = 1);
class Foo {
    private static $format;
    public function __construct($format) {
        static::$format = $format;
    }
    /**
     * @param DateTime $date
     * @return int
     */
    function bar($date) {
        $d = null;
        if ($date === null) $d = $date->format('Y-m-d');
        if ($d)return $d;
        return null;
    }

    function show() {
        echo $this->bar('now');
    }
}

$formats = ['x' => 'Y-m-d'];
(new Foo($formats['y']))->show();
```

Level: 3



```php
<?php declare(strict_types = 1);
class Foo {
    private $format;
    public function __construct($format) {
        $this->format = $format;
    }
    /**
     * @param DateTime $date
     * @return int
     */
    function bar($date) {
        $d = null;
        if ($date === null) $d = $date->format('Y-m-d');
        if ($d)return $d;
        return null;
    }

    function show() {
        echo $this->bar('now');
    }
}

$formats = ['x' => 'Y-m-d'];
(new Foo($formats['y']))->show();
```

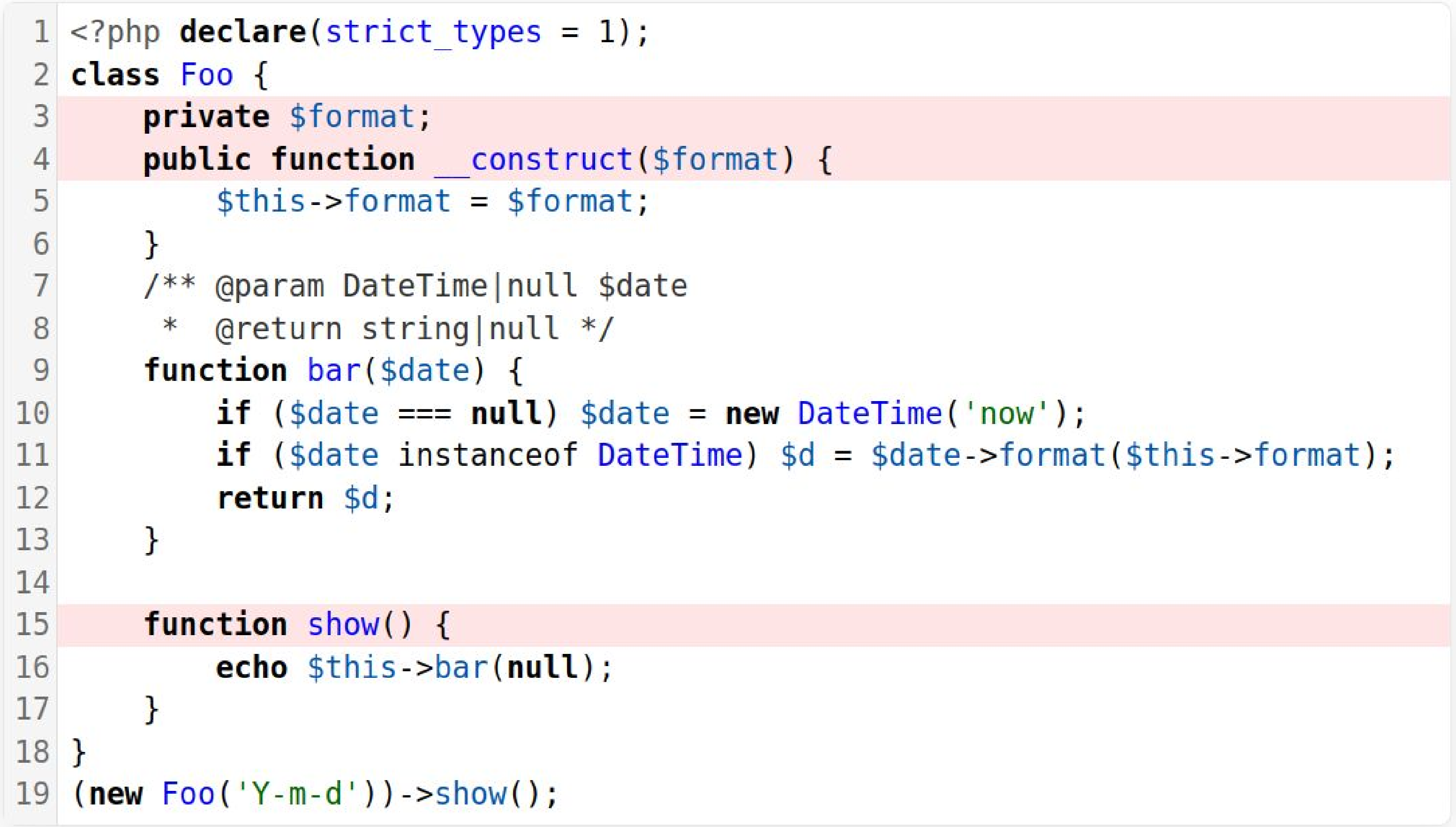
Level: 4

```php
<?php declare(strict_types = 1);
class Foo {
    private $format;
    public function __construct($format) {
        $this->format = $format;
    }
    /** @param DateTime $date
     *  @return string|null */
    function bar($date) {
        $d = null;
        if ($date === null) $d = $date->format('Y-m-d');
        if ($d) return $d;
        return $d;
    }

    function show() {
        echo $this->bar('now');
    }
}
(new Foo('Y-m-d'))->show();
```

Level:
5

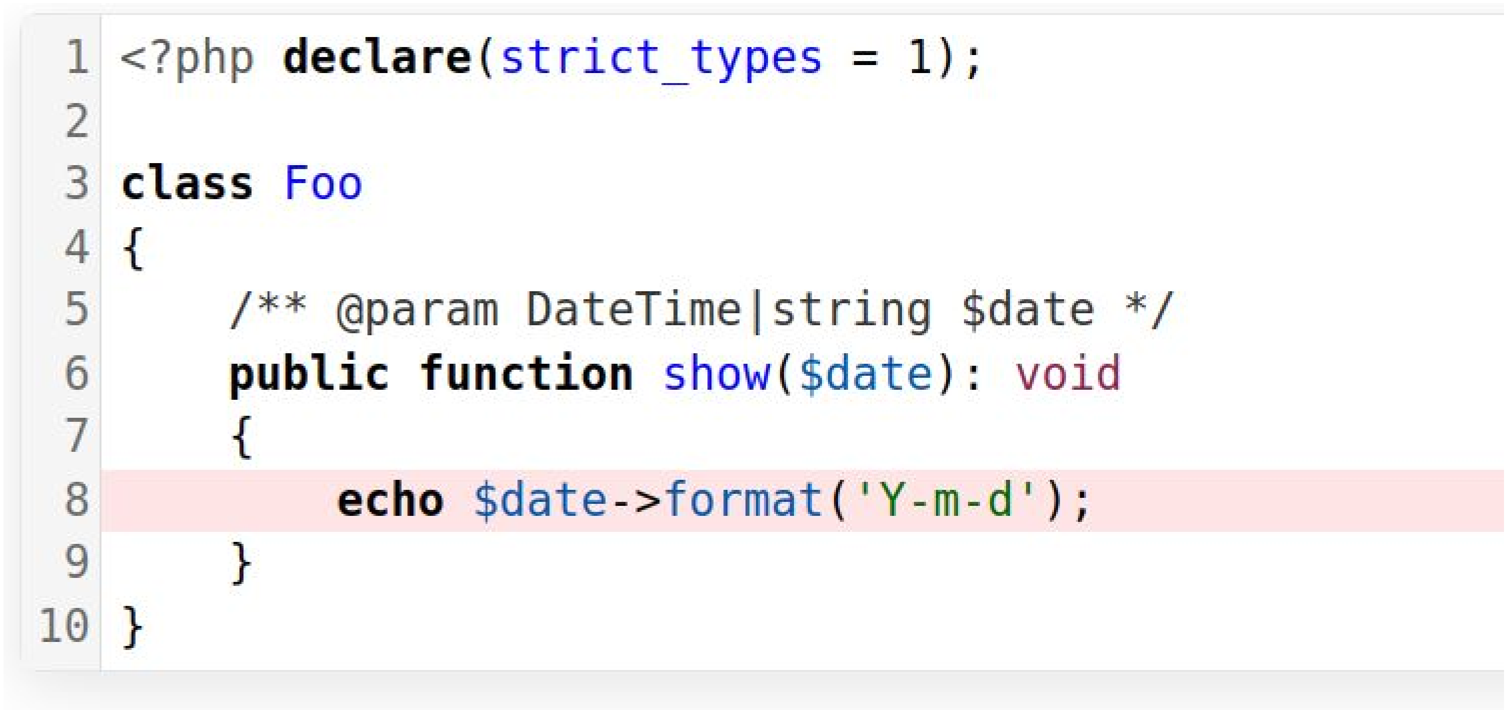
```php
<?php declare(strict_types = 1);
class Foo {
    private $format;
    public function __construct($format) {
        $this->format = $format;
    }
    /** @param DateTime|null $date
     *  @return string|null */
    function bar($date) {
        $d = null;
        if ($date instanceof DateTime) $d = $date->format($this->format);
        return $d;
    }

    function show() {
        echo $this->bar('now');
    }
}
(new Foo('Y-m-d'))->show();
```

Level: 6



```php
<?php declare(strict_types = 1);
class Foo {
    private $format;
    public function __construct($format) {
        $this->format = $format;
    }
    /** @param DateTime|null $date
     *  @return string|null */
    function bar($date) {
        if ($date === null) $date = new DateTime('now');
        if ($date instanceof DateTime) $d = $date->format($this->format);
        return $d;
    }

    function show() {
        echo $this->bar(null);
    }
}
(new Foo('Y-m-d'))->show();
```

Level:
7

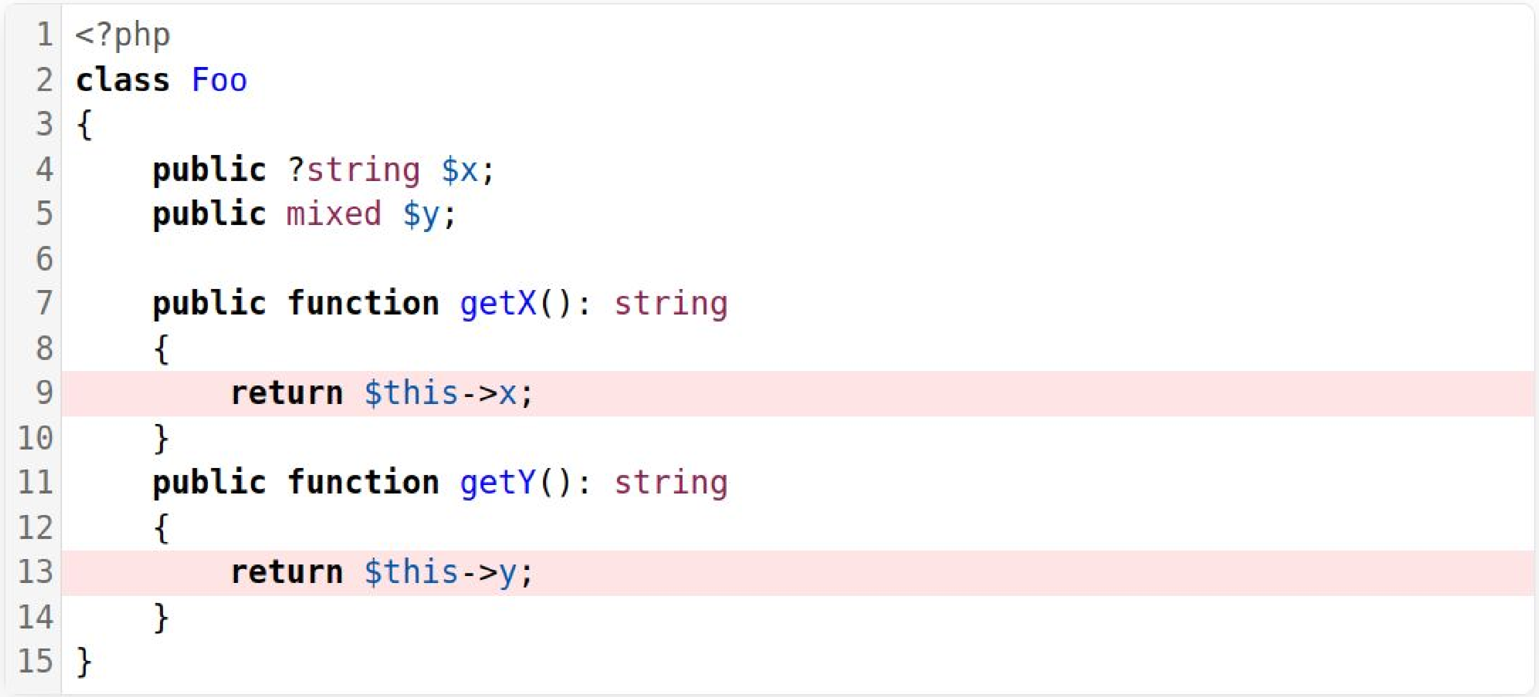


```php
<?php declare(strict_types = 1);

class Foo
{
    /** @param DateTime|string $date */
    public function show($date): void
    {
        echo $date->format('Y-m-d');
    }
}
```

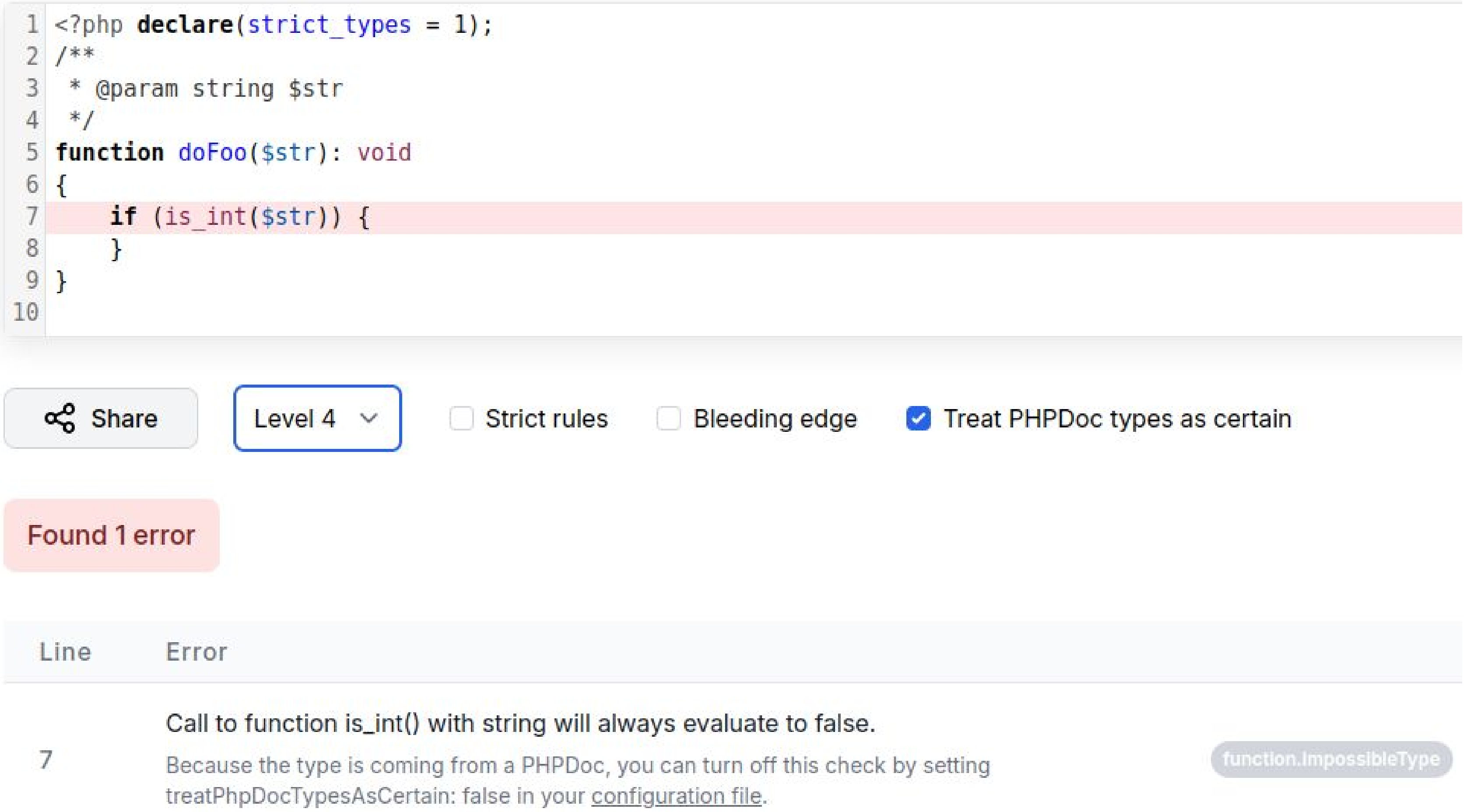
Level: 8

```php
<?php declare(strict_types = 1);
class Foo {
    private ?string $format;
    public function __construct(?string $format) {
        $this->format = $format;
    }
    /** @param DateTime|null $date
     *  @return string|null */
    function bar($date = null) {
        if ($date === null) $date = new DateTime('now');
        if ($date instanceof DateTime) $d = $date->format($this->format);
        return $d;
    }

    function show(): void {
        echo $this->bar();
    }
}
(new Foo('Y-m-d'))->show();
```

Level:
9



```php
<?php
class Foo
{
    public ?string $x;
    public mixed $y;

    public function getX(): string
    {
        return $this->x;
    }
    public function getY(): string
    {
        return $this->y;
    }
}
```

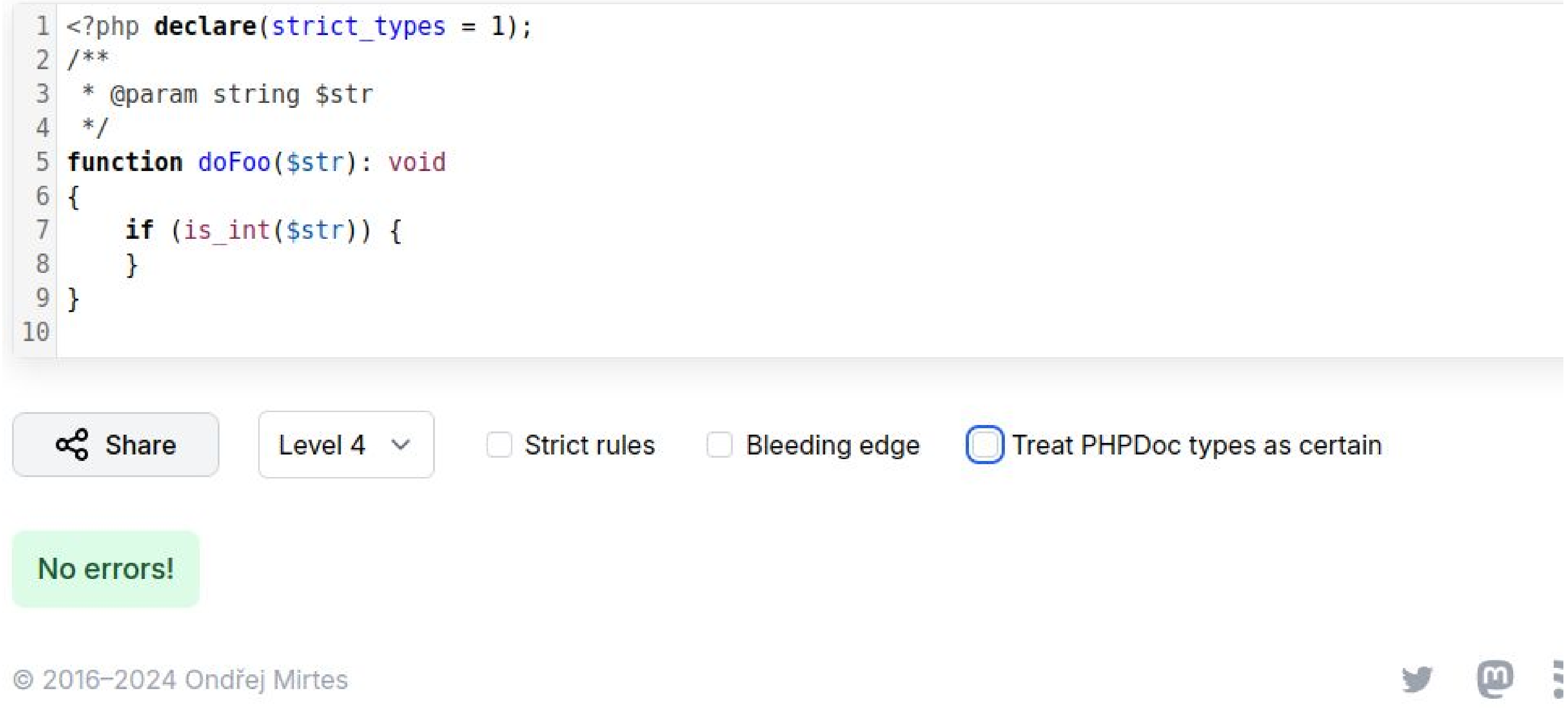


```php
<?php declare(strict_types = 1);
/**
 * @param string $str
 */
function doFoo($str): void
{
    if (is_int($str)) {
    }
}
```

[ ] Strict rules [ ] Bleeding edge [✓] Treat PHPDoc types as certain

Found 1 error

Line	Error
7	Call to function is_int() with string will always evaluate to false. Because the type is coming from a PHPDoc, you can turn off this check by setting treatPhpDocTypesAsCertain: false in your configuration file. function.impossibleType



```php
<?php declare(strict_types = 1);
/**
 * @param string $str
 */
function doFoo($str): void
{
    if (is_int($str)) {
    }
}

```

Strict rules [ ]   Bleeding edge [ ]   Treat PHPDoc types as certain [ ]

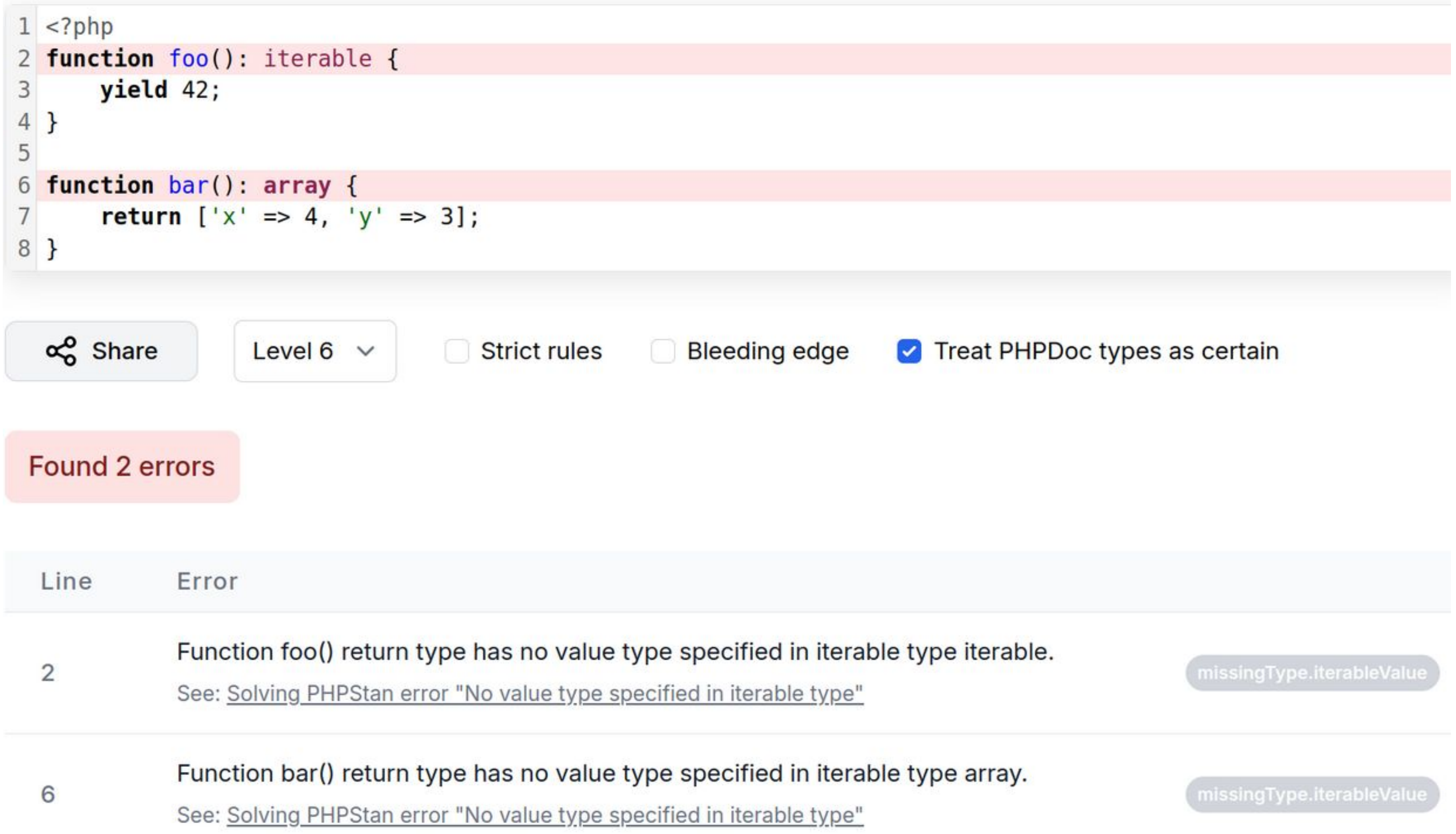
No errors!

`bleeding-edge` 😈

`strict-rules` 😈

`missingCheckedExceptionInThrows: true` 😈😈

Quel niveau est le mieux ?

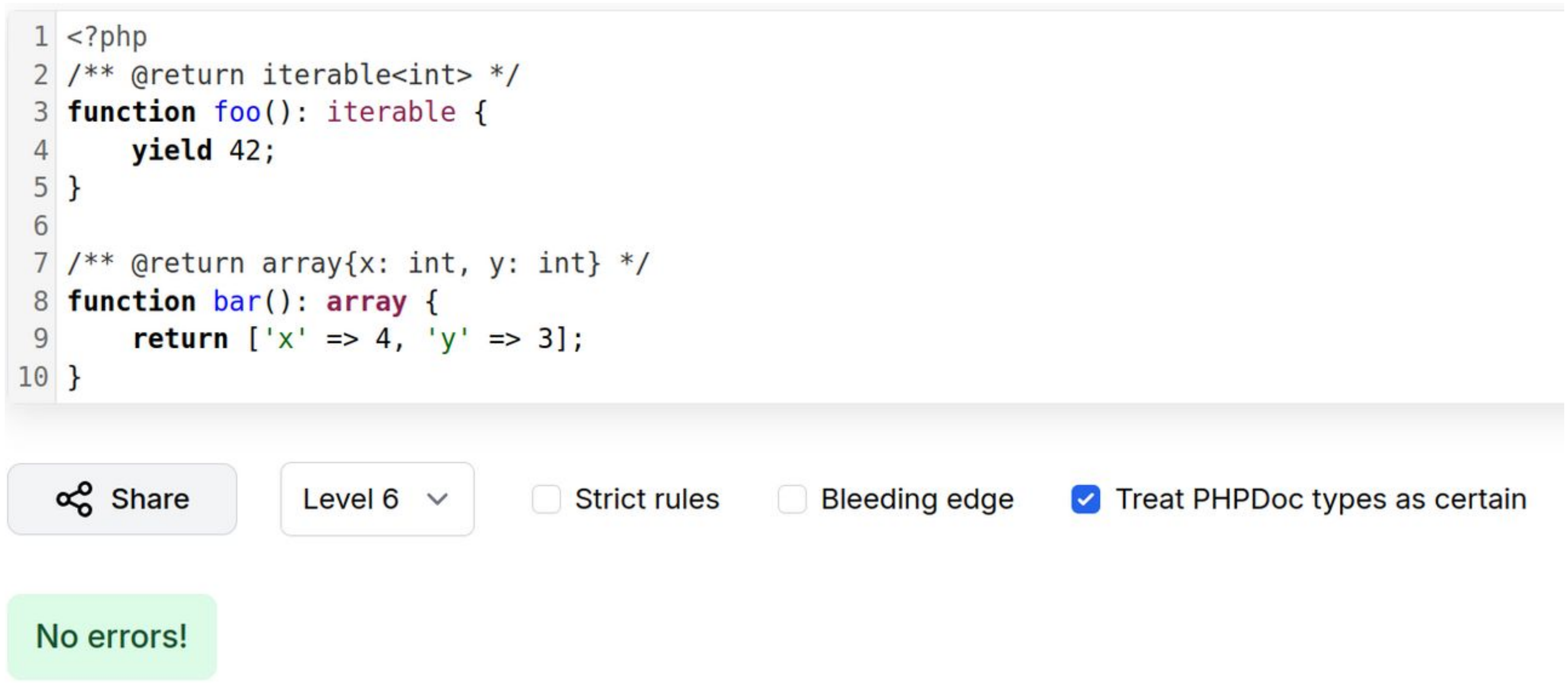


```php
<?php
function foo(): iterable {
    yield 42;
}

function bar(): array {
    return ['x' => 4, 'y' => 3];
}
```

Share | Level 6 | ☐ Strict rules | ☐ Bleeding edge | ☑ Treat PHPDoc types as certain

Found 2 errors

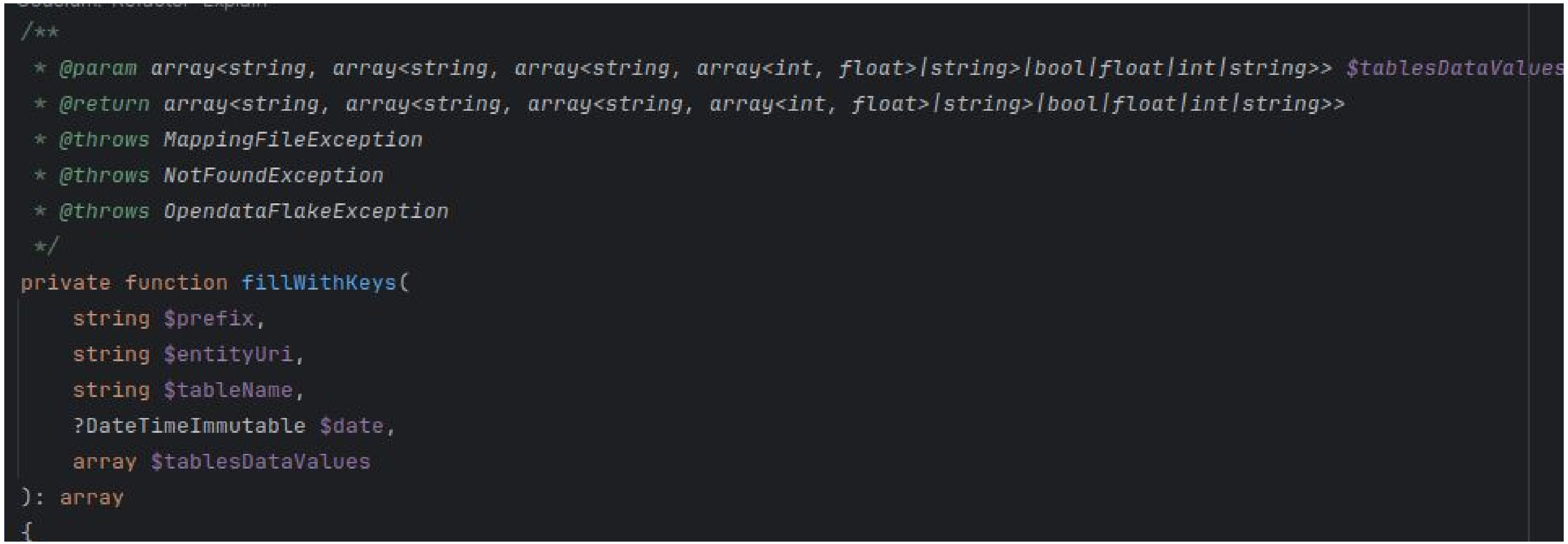
Line	Error	
2	Function foo() return type has no value type specified in iterable type iterable. See: Solving PHPStan error "No value type specified in iterable type"	missingType.iterableValue
6	Function bar() return type has no value type specified in iterable type array. See: Solving PHPStan error "No value type specified in iterable type"	missingType.iterableValue



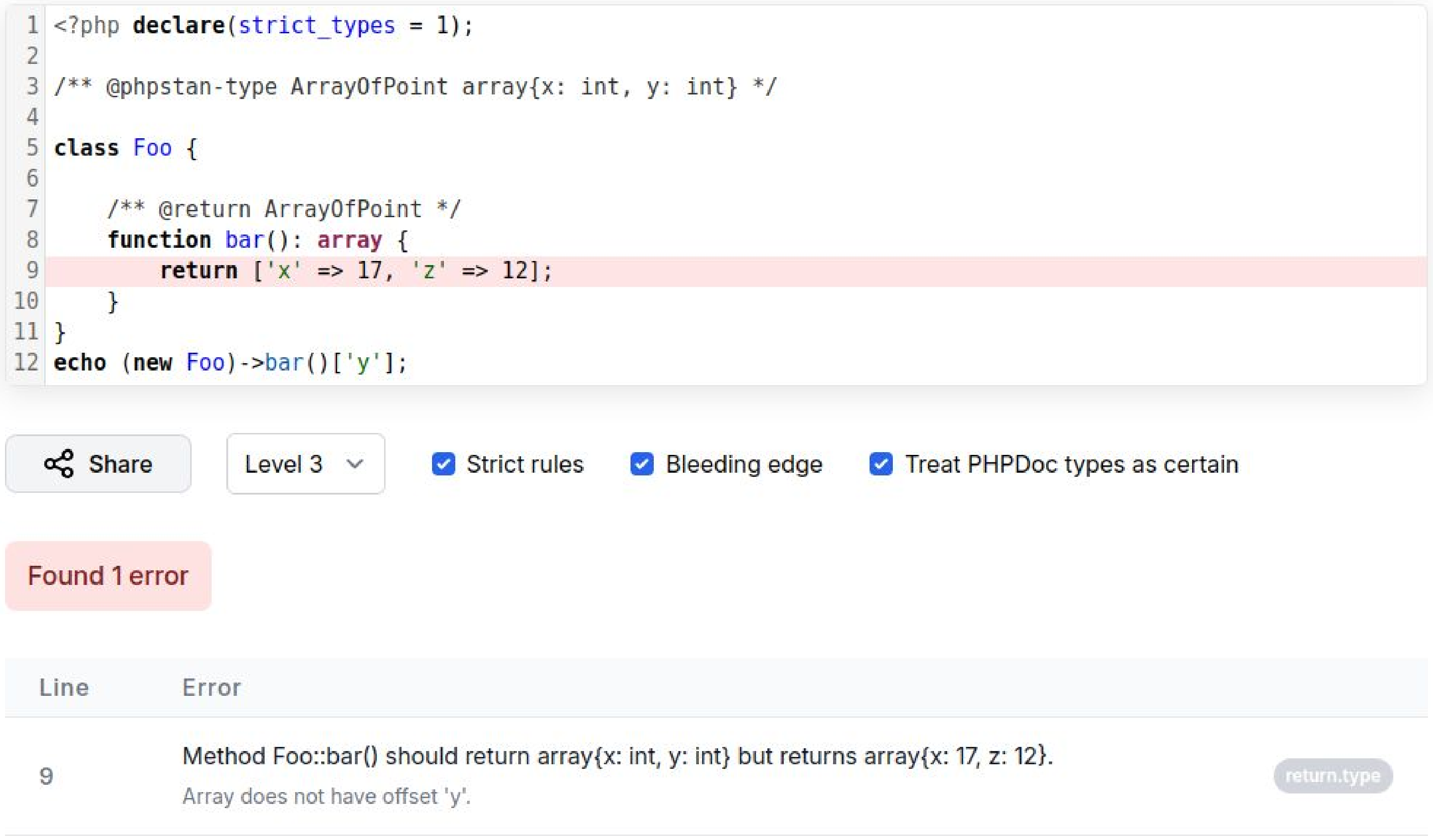
```php
<?php
/** @return iterable<int> */
function foo(): iterable {
    yield 42;
}

/** @return array{x: int, y: int} */
function bar(): array {
    return ['x' => 4, 'y' => 3];
}
```

[ ] Strict rules [ ] Bleeding edge [✓ Treat PHPDoc types as certain]

No errors!



```
/**
 * @param array<string, array<string, array<string, array<int, float>|string>|bool|float|int|string>> $tablesDataValues
 * @return array<string, array<string, array<string, array<int, float>|string>|bool|float|int|string>>
 * @throws MappingFileException
 * @throws NotFoundException
 * @throws OpendataFlakeException
 */
private function fillWithKeys(
    string $prefix,
    string $entityUri,
    string $tableName,
    ?DateTimeImmutable $date,
    array $tablesDataValues
): array
{
```

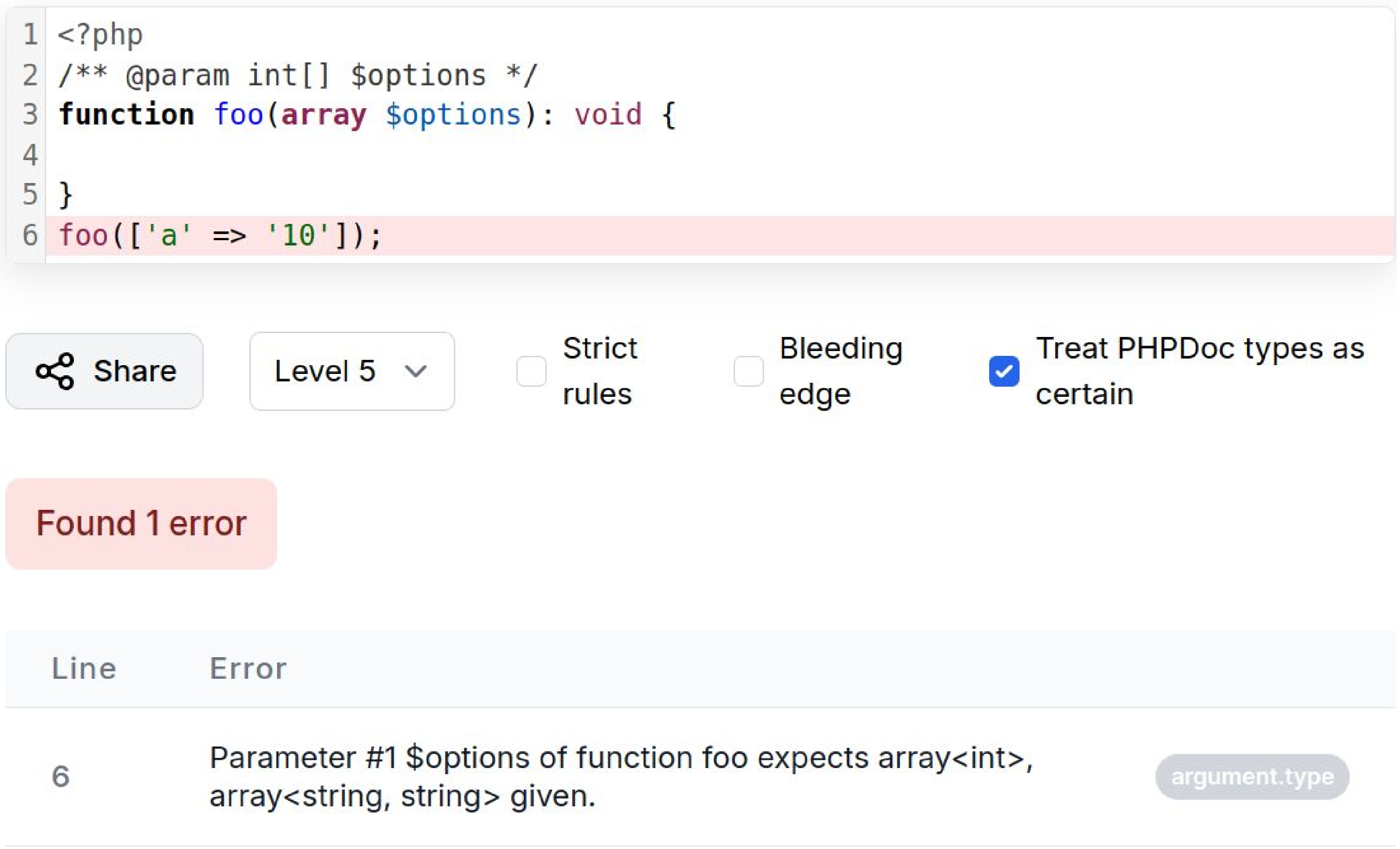


```php
<?php declare(strict_types = 1);

/** @phpstan-type ArrayOfPoint array{x: int, y: int} */

class Foo {

    /** @return ArrayOfPoint */
    function bar(): array {
        return ['x' => 17, 'z' => 12];
    }
}
echo (new Foo)->bar()['y'];
```

Share | Level 3 | ✓ Strict rules | ✓ Bleeding edge | ✓ Treat PHPDoc types as certain

Found 1 error

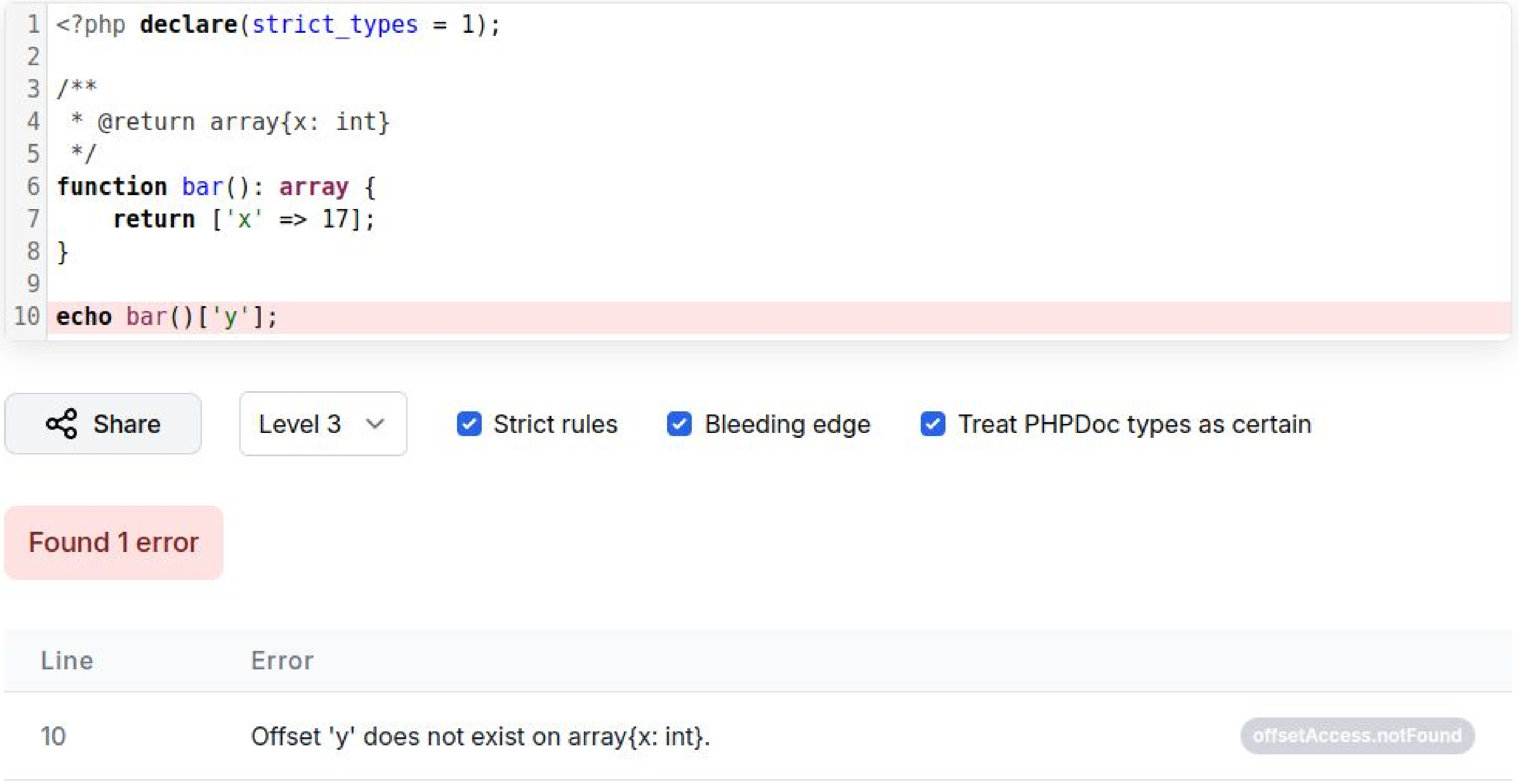
Line	Error	
9	Method Foo::bar() should return array{x: int, y: int} but returns array{x: 17, z: 12}. Array does not have offset 'y'.	return.type



```php
<?php
/** @param int[] $options */
function foo(array $options): void {

}
foo(['a' => '10']);
```

Found 1 error

Line	Error	
6	Parameter #1 $options of function foo expects array<int>, array<string, string> given.	argument.type



```php
<?php declare(strict_types = 1);

/**
 * @return array{x: int}
 */
function bar(): array {
    return ['x' => 17];
}

echo bar()['y'];
```

Found 1 error

Line	Error	
10	Offset 'y' does not exist on array{x: int}.	offsetAccess.notFound

Appliquons cela sur notre legacy !

Essayons avec Symfony 7.1

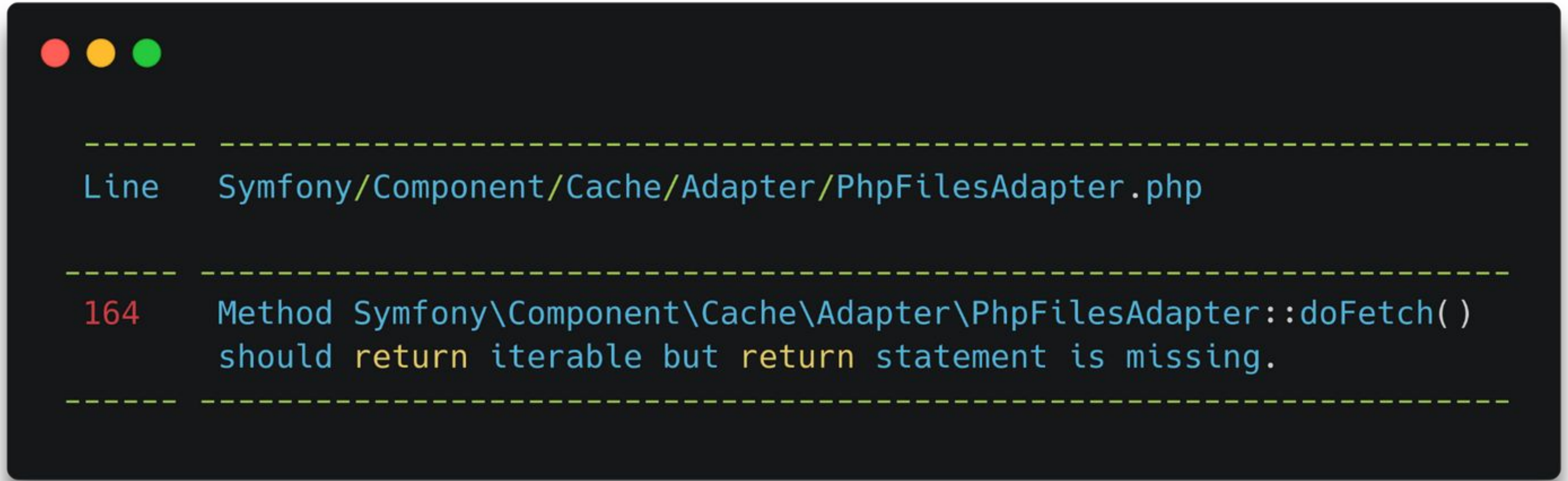
Nombre de fichiers analysés : **5.163**

Nombre de fichiers analysés : **5.163**

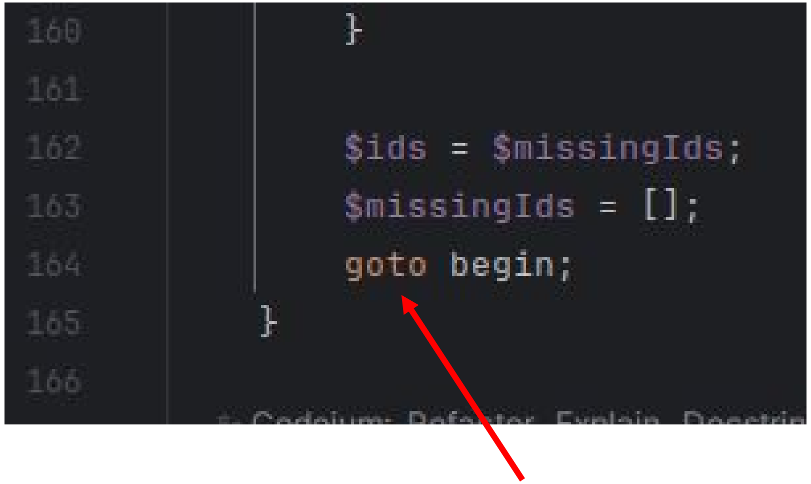
Erreurs phpstan: **16.149** 😱

Nombre de fichiers analysés : **5.163**

Erreurs phpstan level 0: **371**

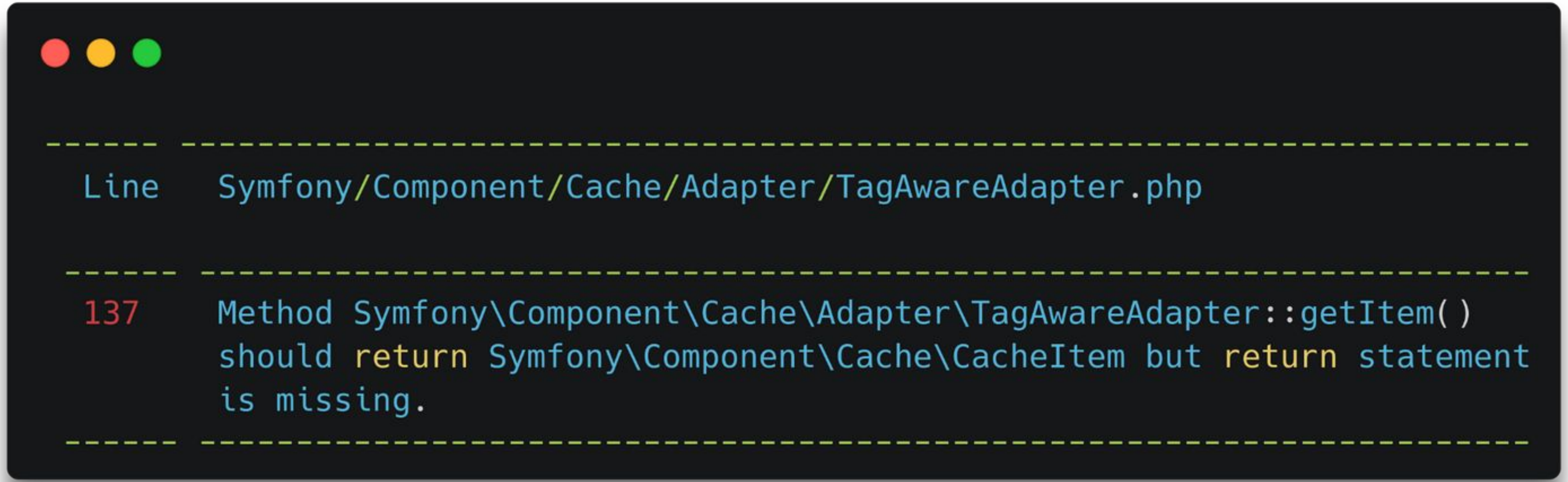


```
 ------ -----------------------------------------------------------
  Line   Symfony/Component/Cache/Adapter/PhpFilesAdapter.php
 ------ -----------------------------------------------------------
  164    Method Symfony\Component\Cache\Adapter\PhpFilesAdapter::doFetch()
         should return iterable but return statement is missing.
 ------ -----------------------------------------------------------
```

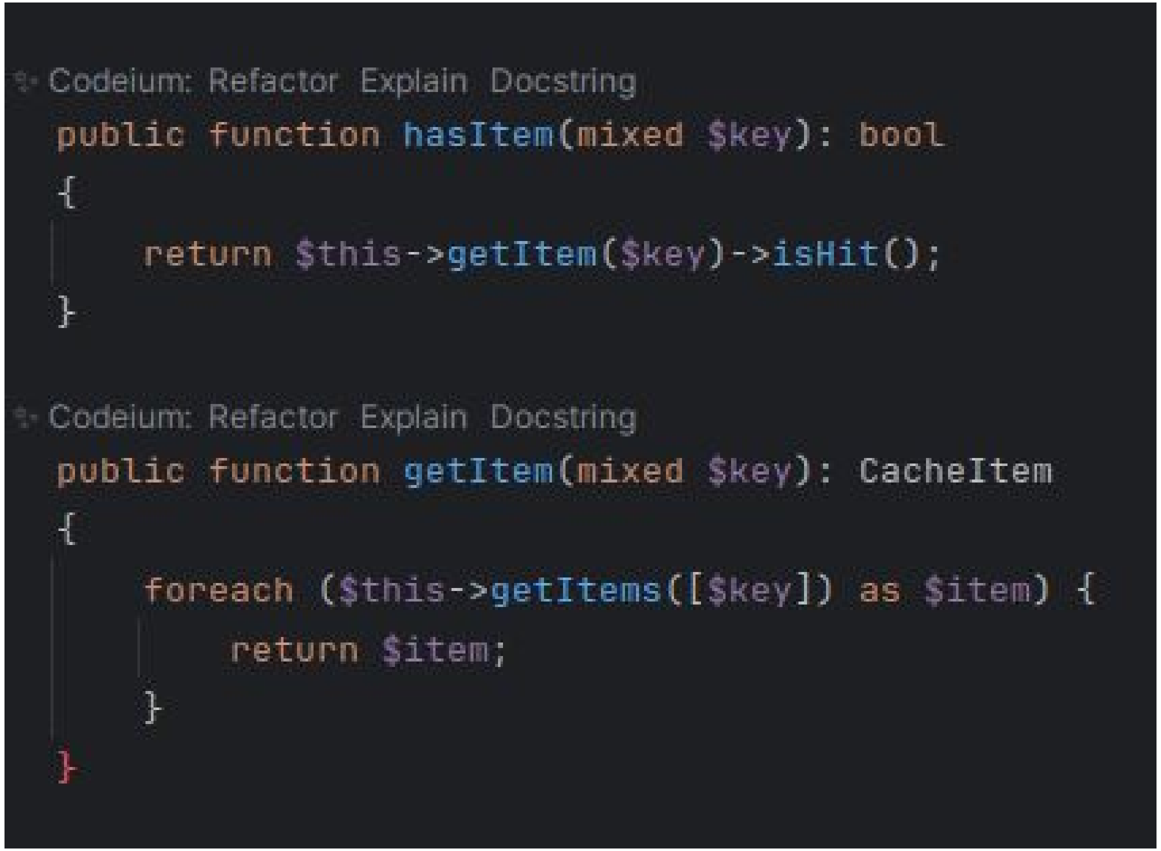


```
        }

        $ids = $missingIds;
        $missingIds = [];
        goto begin;
    }

```

😱



```
 ------ ----------------------------------------------------------------
  Line   Symfony/Component/Cache/Adapter/TagAwareAdapter.php
 ------ ----------------------------------------------------------------
  137    Method Symfony\Component\Cache\Adapter\TagAwareAdapter::getItem()
         should return Symfony\Component\Cache\CacheItem but return statement
         is missing.
 ------ ----------------------------------------------------------------
```


Codeium: Refactor Explain Docstring

```
public function hasItem(mixed $key): bool
{
    return $this->getItem($key)->isHit();
}
```

Codeium: Refactor Explain Docstring

```
public function getItem(mixed $key): CacheItem
{
    foreach ($this->getItems([$key]) as $item) {
        return $item;
    }
}
```

Drupal/core (lib et modules)

Nombre de fichiers analysés : **4.604**

Erreurs phpstan: **1.189**

Drupal/core (lib et modules)

Nombre de fichiers analysés : **4.604**

Erreurs phpstan: **1.189**

Level .....
0 😱

Drupal/core (lib et modules)

Nombre de fichiers analysés : **4.604**

Erreurs phpstan: **1.189**

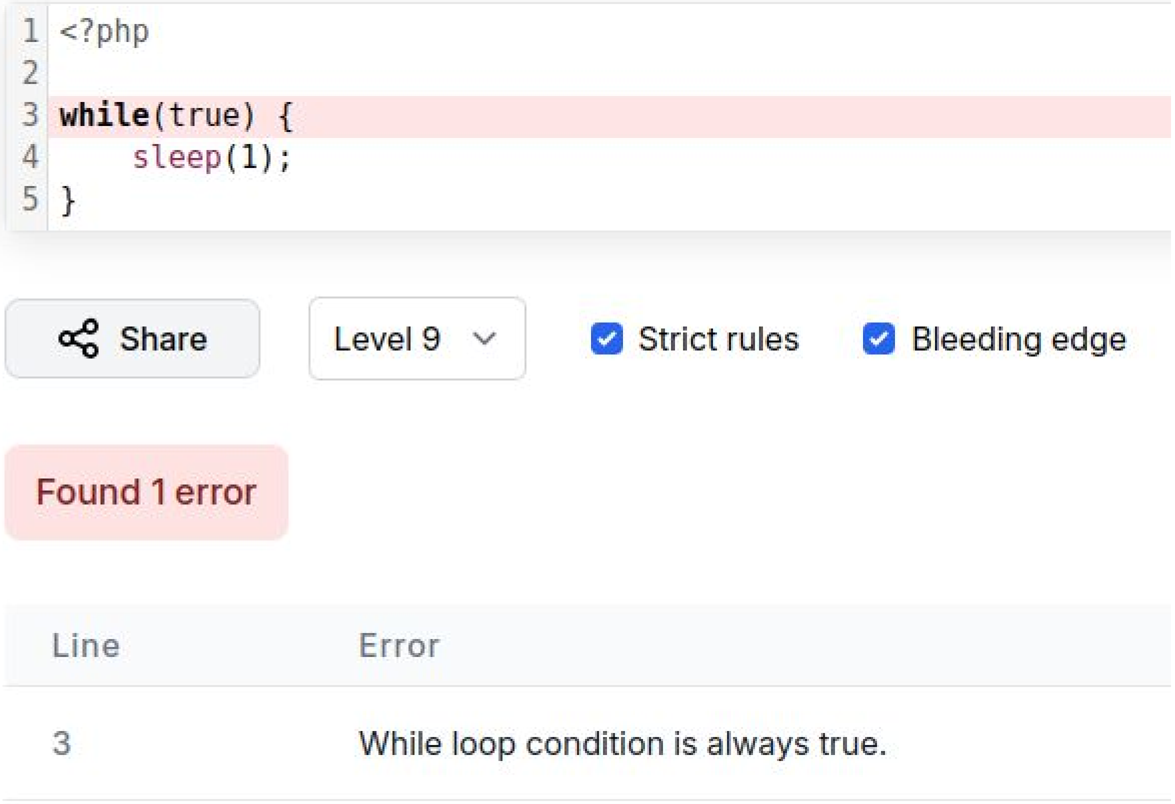
Level .....
0 😱

Level: **9** => **35.736** 😱😱

On laisse tomber ?

Level 0, puis 1, puis 2, ...

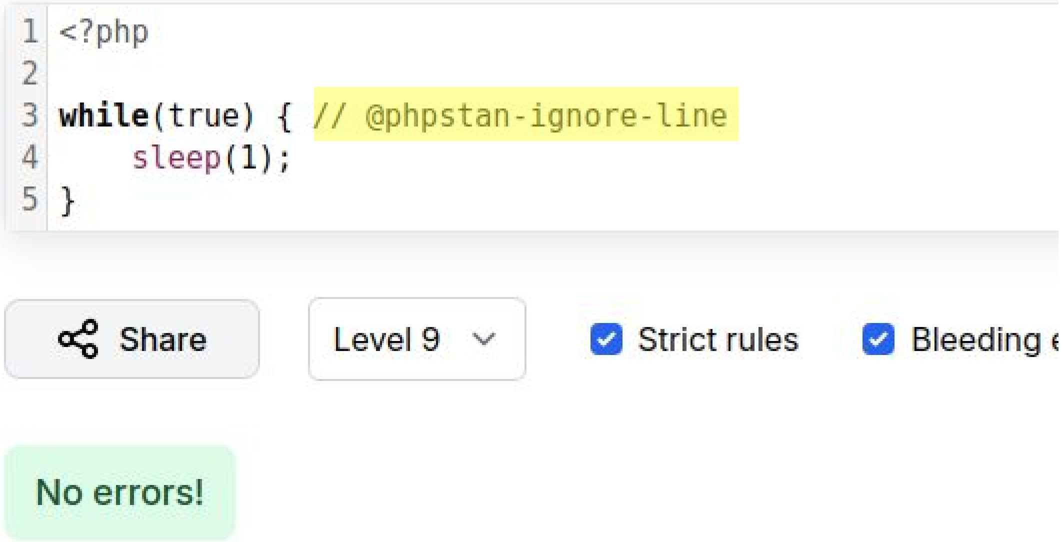
https://phpstan.org/try



```php
<?php

while(true) {
    sleep(1);
}
```

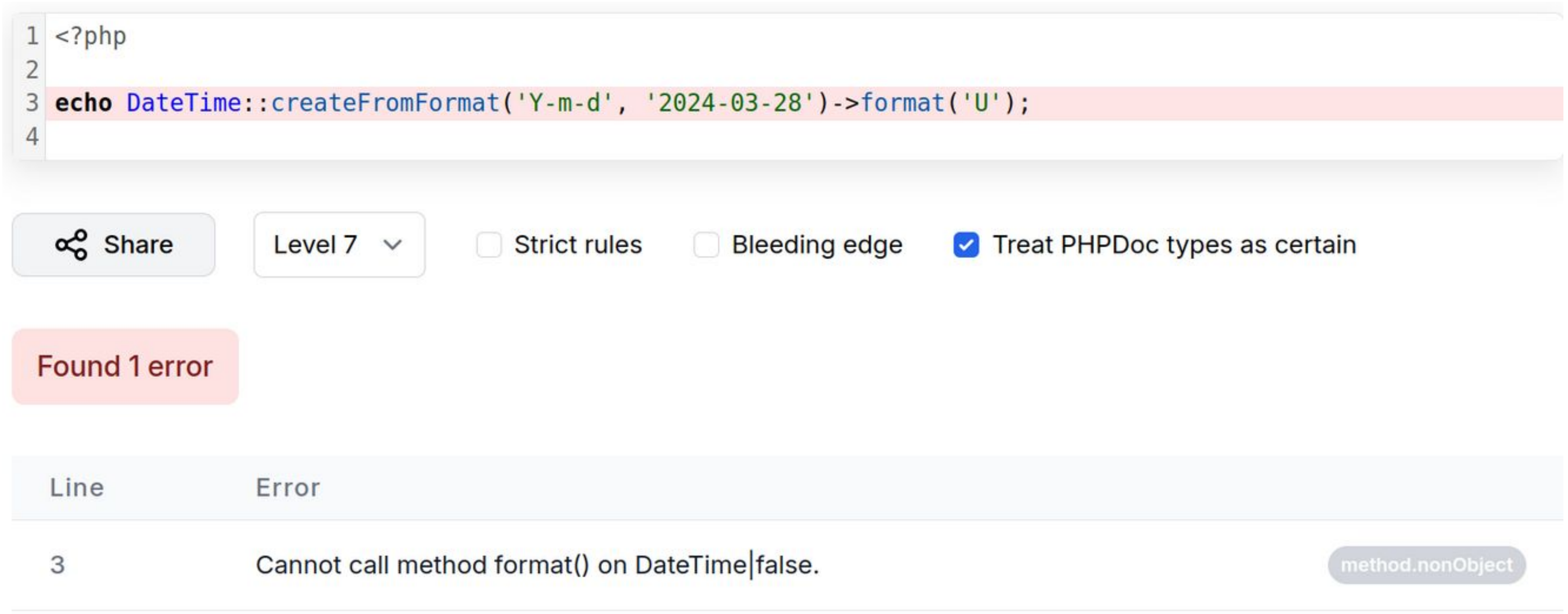
Found 1 error

Line	Error
3	While loop condition is always true.



```php
<?php

while(true) { // @phpstan-ignore-line
    sleep(1);
}
```

No errors!

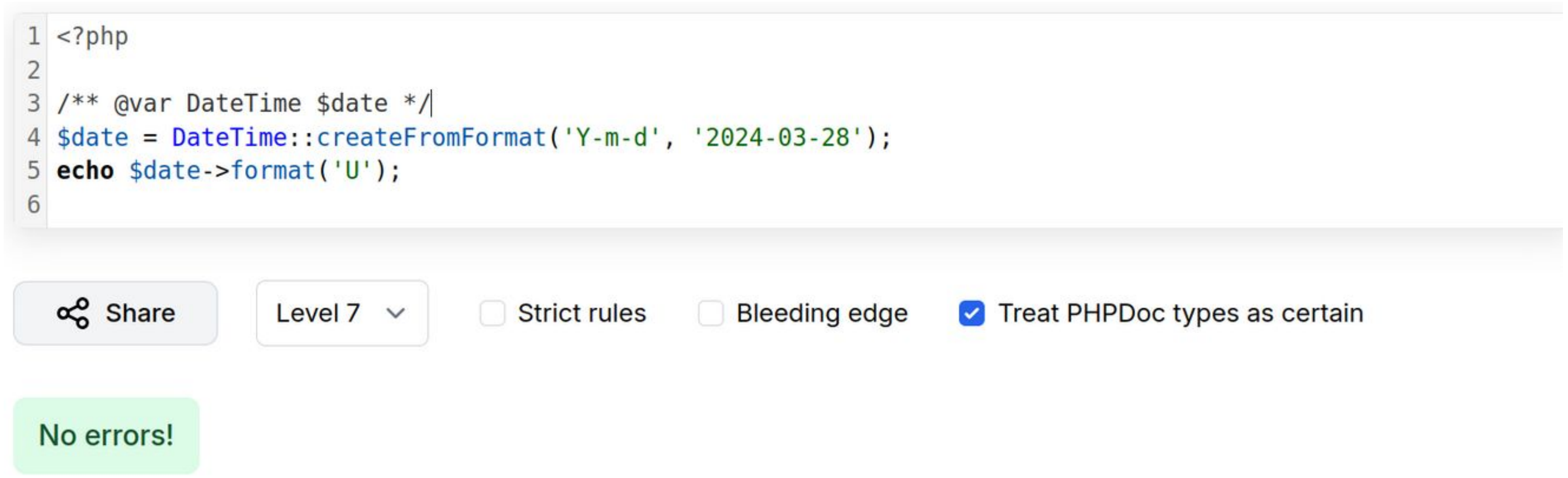


```php
<?php

echo DateTime::createFromFormat('Y-m-d', '2024-03-28')->format('U');

```

Share | Level 7 | ☐ Strict rules | ☐ Bleeding edge | ☑ Treat PHPDoc types as certain

Found 1 error

Line	Error	
3	Cannot call method format() on DateTime|false.	method.nonObject

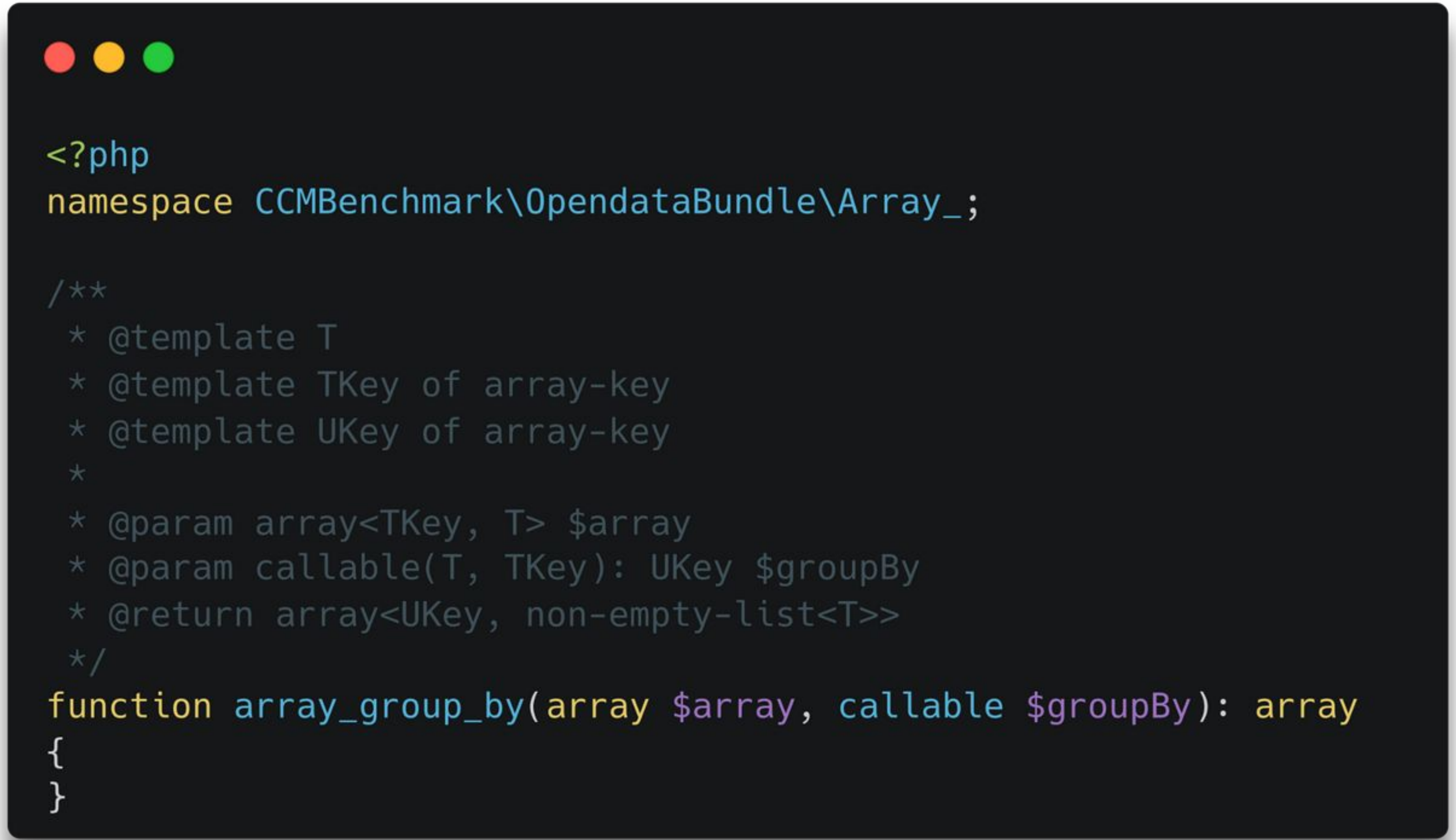


```php
<?php

/** @var DateTime $date */
$date = DateTime::createFromFormat('Y-m-d', '2024-03-28');
echo $date->format('U');

```

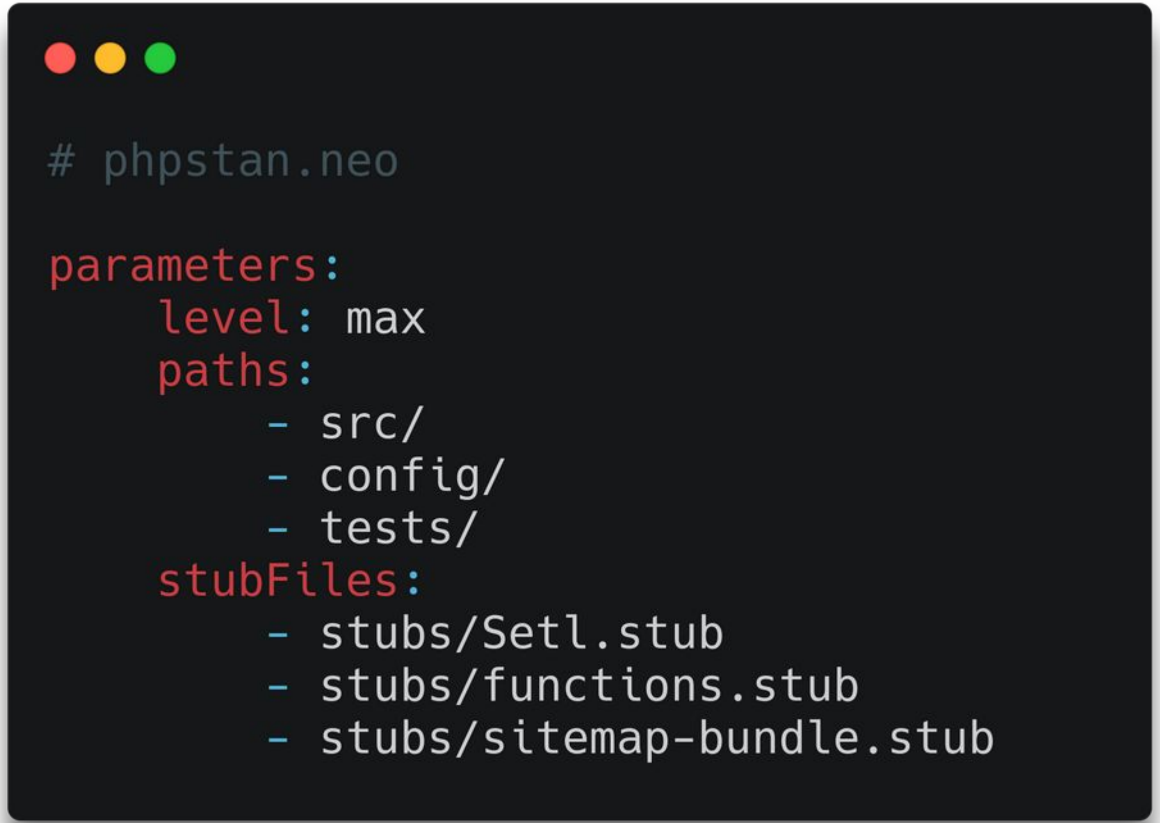
Level 7

[ ] Strict rules [ ] Bleeding edge [✓] Treat PHPDoc types as certain

No errors!



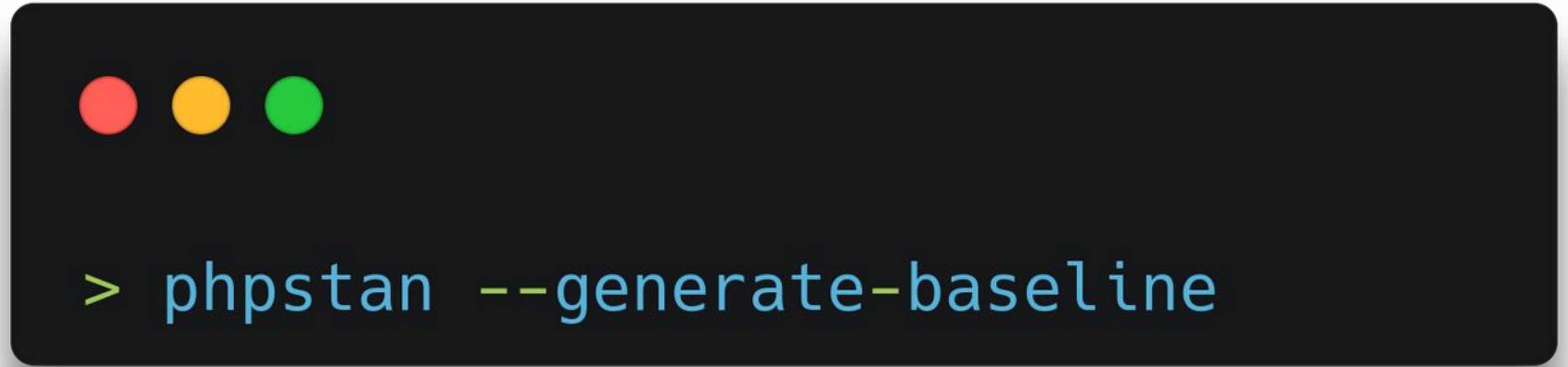
```php
<?php
namespace CCMBenchmark\OpendataBundle\Array_;

/**
 * @template T
 * @template TKey of array-key
 * @template UKey of array-key
 *
 * @param array<TKey, T> $array
 * @param callable(T, TKey): UKey $groupBy
 * @return array<UKey, non-empty-list<T>>
 */
function array_group_by(array $array, callable $groupBy): array
{
}
```



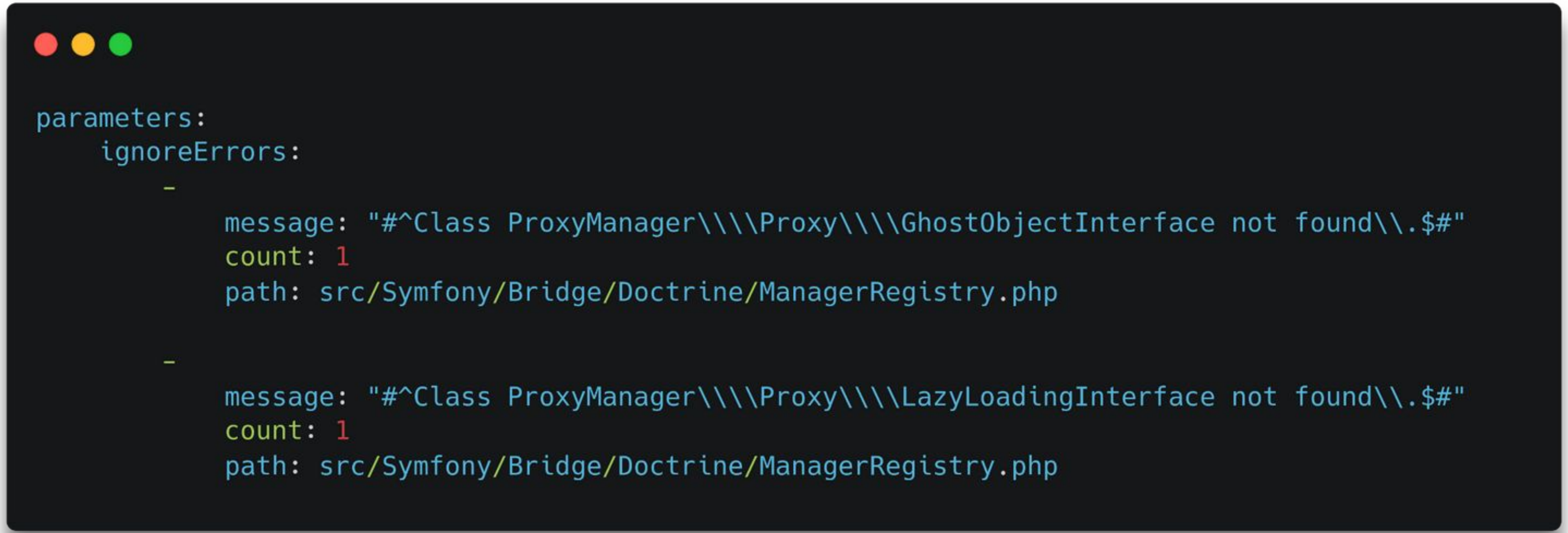
```
# phpstan.neo

parameters:
    level: max
    paths:
        - src/
        - config/
        - tests/
    stubFiles:
        - stubs/Setl.stub
        - stubs/functions.stub
        - stubs/sitemap-bundle.stub
```

“ Pas d’erreur en level 0 “

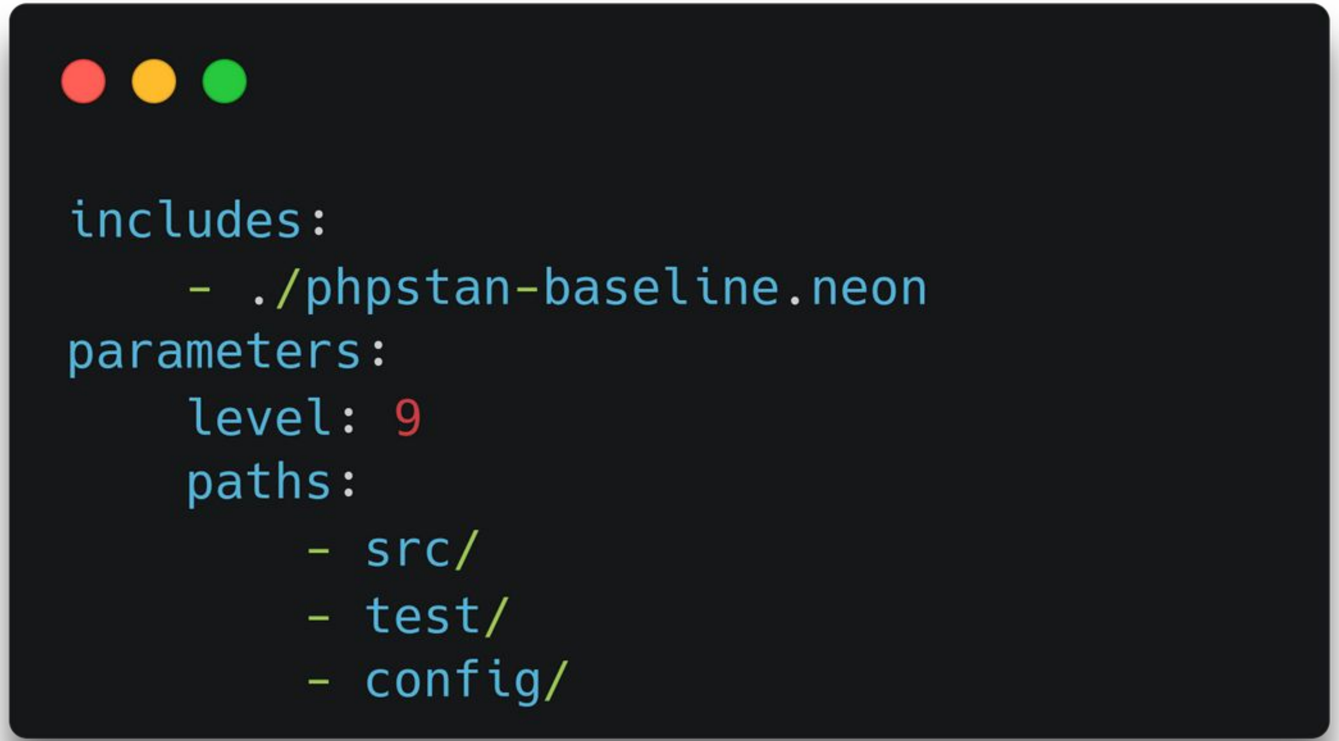
Solution : la baseline !



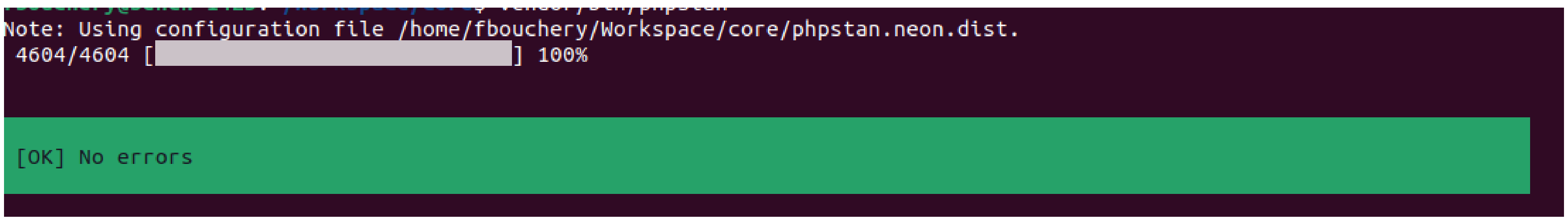
```
> phpstan --generate-baseline
```



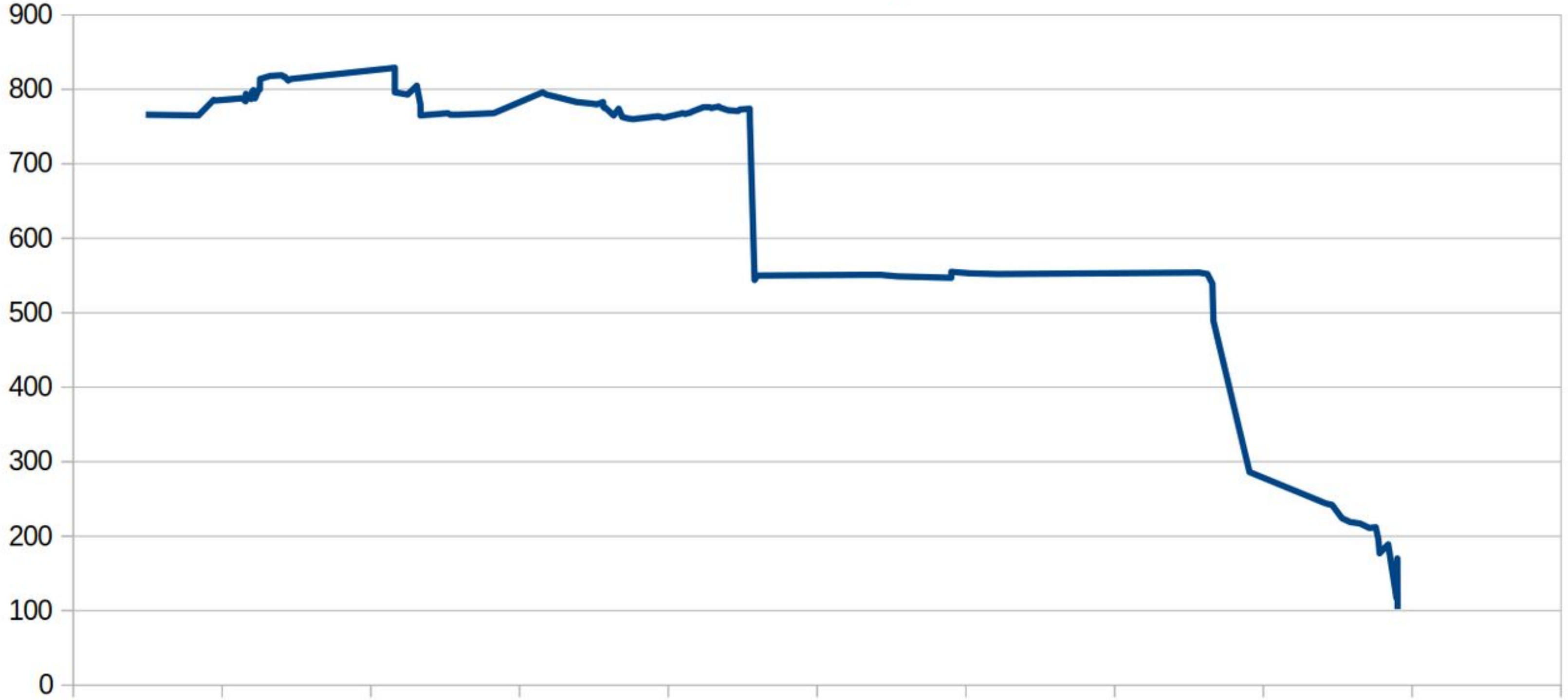
```yaml
parameters:
    ignoreErrors:
        -
            message: "#^Class ProxyManager\\\\Proxy\\\\GhostObjectInterface not found\\.$#"
            count: 1
            path: src/Symfony/Bridge/Doctrine/ManagerRegistry.php

        -
            message: "#^Class ProxyManager\\\\Proxy\\\\LazyLoadingInterface not found\\.$#"
            count: 1
            path: src/Symfony/Bridge/Doctrine/ManagerRegistry.php
```

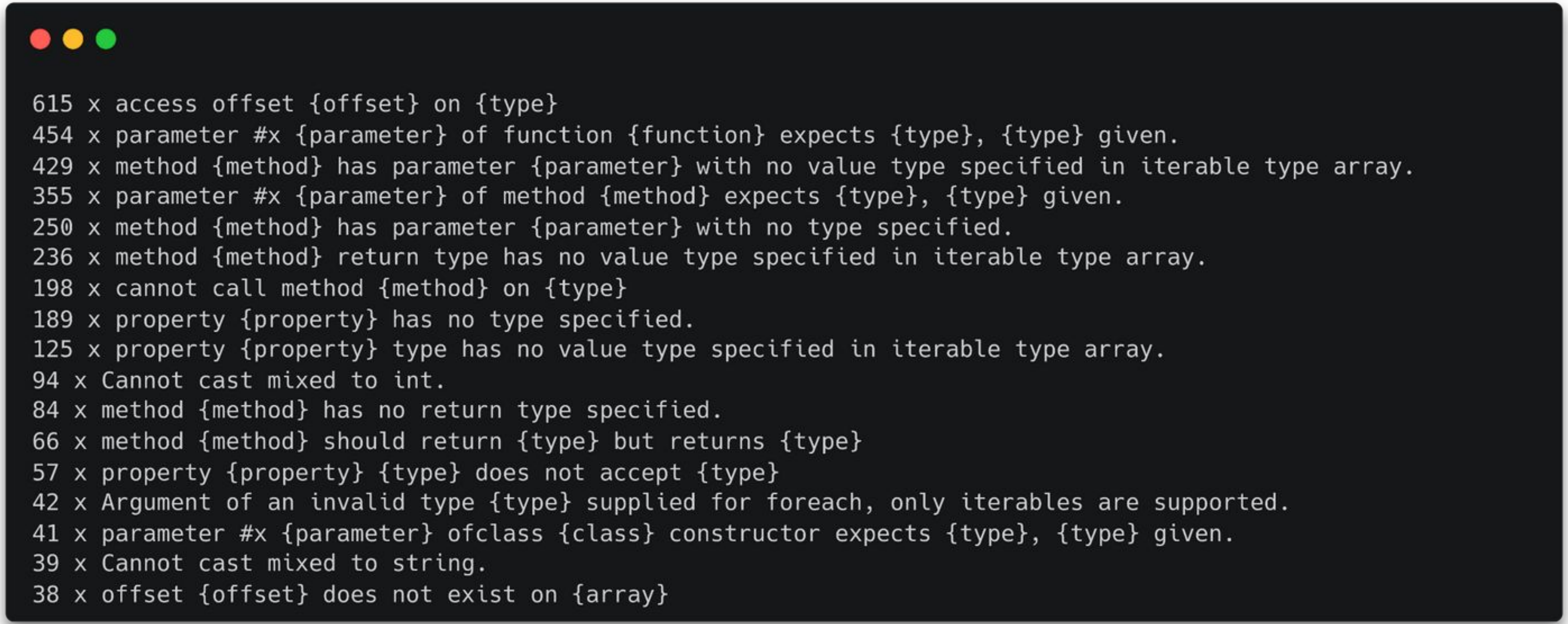


```
includes:
    - ./phpstan-baseline.neon
parameters:
    level: 9
    paths:
        - src/
        - test/
        - config/
```



```
Note: Using configuration file /home/fbouchery/Workspace/core/phpstan.neon.dist.
 4604/4604 [▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓] 100%



 [OK] No errors

```


900
800
700
600
500
400
300
200
100
0



```
615 x access offset {offset} on {type}
454 x parameter #x {parameter} of function {function} expects {type}, {type} given.
429 x method {method} has parameter {parameter} with no value type specified in iterable type array.
355 x parameter #x {parameter} of method {method} expects {type}, {type} given.
250 x method {method} has parameter {parameter} with no type specified.
236 x method {method} return type has no value type specified in iterable type array.
198 x cannot call method {method} on {type}
189 x property {property} has no type specified.
125 x property {property} type has no value type specified in iterable type array.
94 x Cannot cast mixed to int.
84 x method {method} has no return type specified.
66 x method {method} should return {type} but returns {type}
57 x property {property} {type} does not accept {type}
42 x Argument of an invalid type {type} supplied for foreach, only iterables are supported.
41 x parameter #x {parameter} ofclass {class} constructor expects {type}, {type} given.
39 x Cannot cast mixed to string.
38 x offset {offset} does not exist on {array}
```

RECTOR

your high-digital place open

AXESS
Digital & Business Experts

SMILE

Talan

Merci
pour votre écoute !

DRUPALCAMP ROAZHON
Rennes - 28 au 30 mars 2024

@FredBouchery
(sur X, Bluesky ou Mastodon)