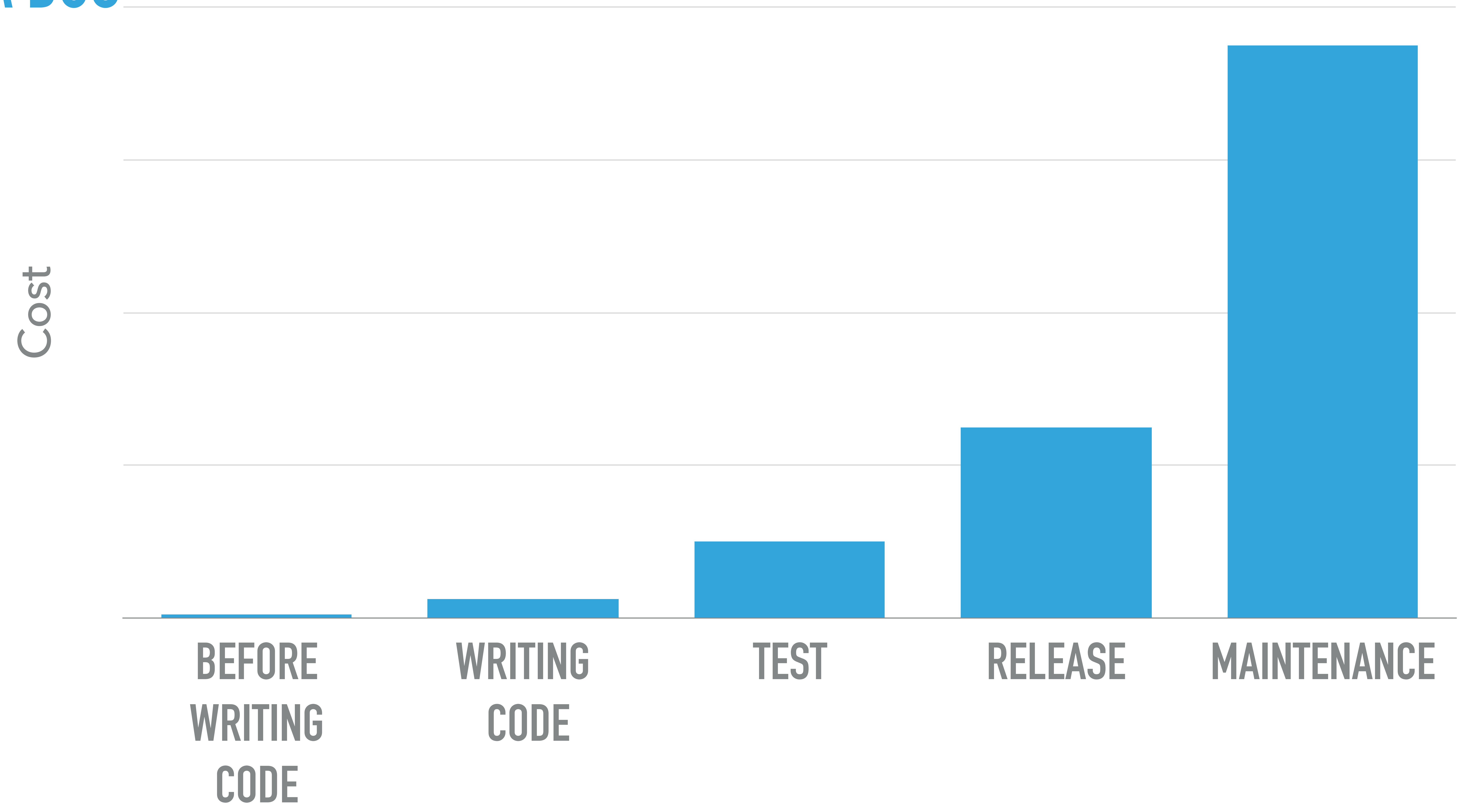
DAVE LIDDAMENT

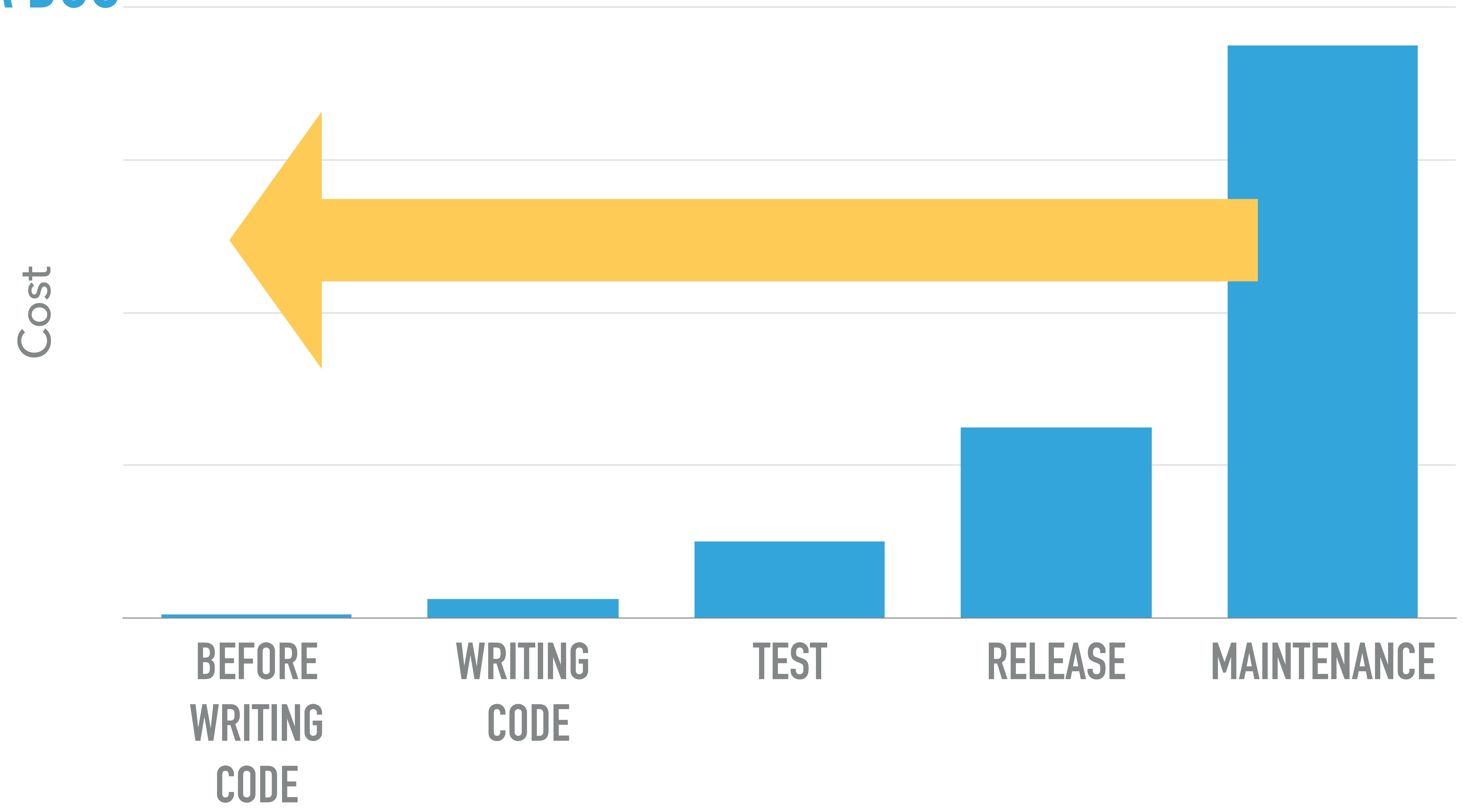
SQUASH BUGS WITH STATIC ANALYSIS

APPROPRIATE APPLICATION OF STATIC ANALYSIS REDUCES THE OVERALL COST OF SOFTWARE DEVELOPMENT.

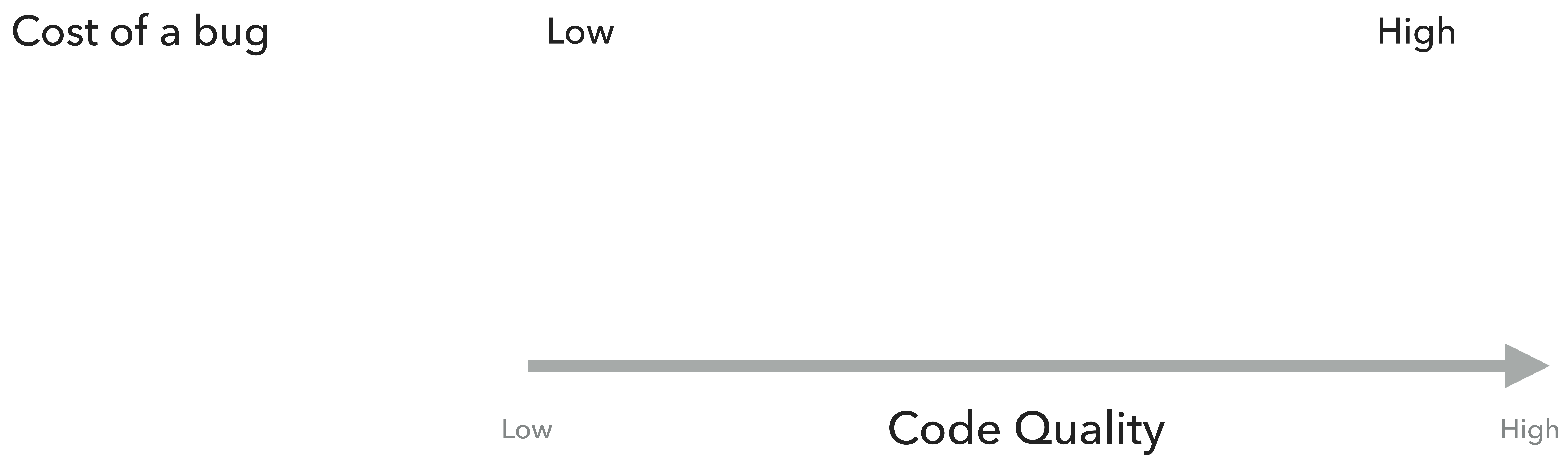
COST OF A BUG


Cost
BEFORE WRITING CODE
WRITING CODE
TEST
RELEASE
MAINTENANCE

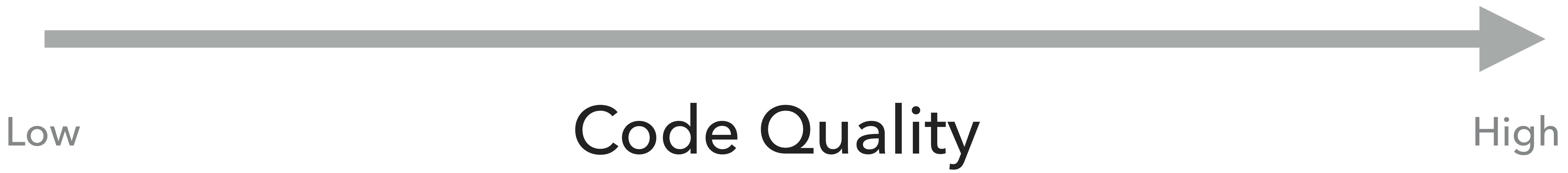
COST OF A BUG


Cost
BEFORE WRITING CODE
WRITING CODE
TEST
RELEASE
MAINTENANCE

Low
Code Quality
High

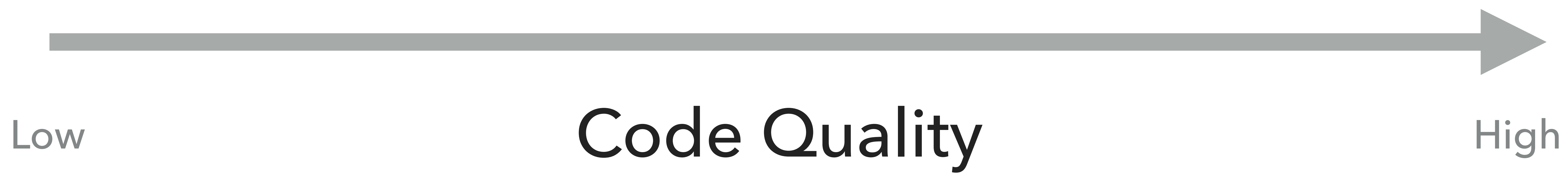

Cost of a bug
Low
High
Low
Code Quality
High

Cost of a bug	Low	High
Project duration	Short	Long



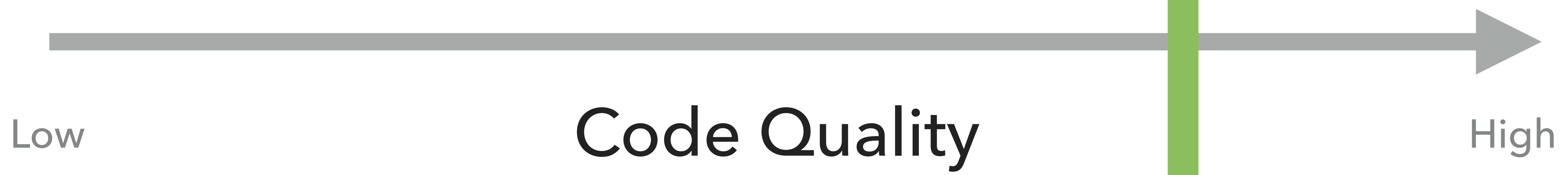
WHY

Cost of a bug	Low	High
Project duration	Short	Long
Subset of PHP used	More	Less



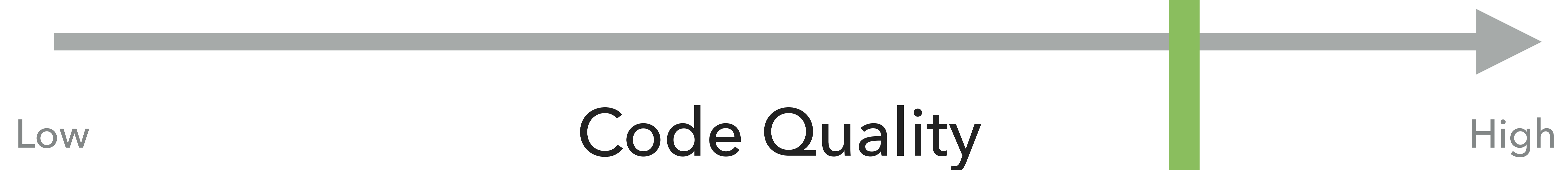
WHY

Cost of a bug	Low	High
Project duration	Short	Long
Subset of PHP used	More	Less



WHY

Cost of a bug	Low	High
Project duration	Short	Long
Subset of PHP used	More	Less



Barrier to entry	Low	High

@daveliddament

WHY

```
function processPerson($person, $age) {
  if ($age == 18) {
    return “You’re 18”;
  }
 if (getManager() == $person) {
    return “You’re a manager”;
  }


}


function getManager(): Person {… some code …}
```

@daveliddament

WHY

```
function processPerson($person, $age) {
  if ($age == 18) {
    return “You’re 18”;
  }
 if (getManager() == $person) {
    return “You’re a manager”;
  }

}

function getManager(): Person {… some code …}
```

WHY

```
function processPerson($person, $age) {
  if ($age == 18) {
    return "You're 18";
  }
 if (getManager() == $person) {
    return "You're a manager";
  }

}

function getManager(): Person {… some code …}
```

WHY

```
function processPerson($person, $age) {
  if ($age == 18) {
    return "You're 18";
  }
 if (getManager() == $person) {
    return "You're a manager";
  }


}


function getManager(): Person {… some code …}
```

@daveliddament

WHY

```
function processPerson($person, $age) {
  if ($age == 18) {
    return "You're 18";
  }
 if (getManager() == $person) {
    return "You're a manager";
  }

}

function getManager(): Person {… some code …}
```

@daveliddament

WHY

```
function processPerson($person, $age) {

  if ($age == 18) {

    return "You're 18";

  }

 if (getManager() == $person) {

    return "You're a manager";

  }


}


function getManager(): Person {… some code …}
```

WHY

```
declare(strict_types=1);

function processPerson(Person $person, int $age): ?string {

  if ($age === 18) {

    return “You’re 18”;

  }

 if (getManager()->isEqual($person)) {

    return “You’re a manager”;

  }

  return null;

}


function getManager(): Person {… some code …}
```

```
declare(strict_types=1);
function processPerson(Person $person, int $age): ?string {
  if ($age === 18) {
    return “You’re 18”;
  }
 if (getManager()->isEqual($person)) {
    return “You’re a manager”;
  }
  return null;
}


function getManager(): Person {… some code …}
```

WHY

```
declare(strict_types=1);

function processPerson(Person $person, int $age): ?string {

  if ($age === 18) {

    return “You’re 18”;

  }

 if (getManager()->isEqual($person)) {

    return “You’re a manager”;

  }

  return null;

}


function getManager(): Person {… some code …}
```

```
declare(strict_types=1);

function processPerson(Person $person, int $age): ?string {

  if ($age === 18) {

    return "You're 18";

  }

 if (getManager()->isEqual($person)) {

    return "You're a manager";

  }

  return null;

}


function getManager(): Person {… some code …}
```

```
declare(strict_types=1);

function processPerson(Person $person, int $age): ?string {

  if ($age === 18) {

    return “You’re 18”;

  }

 if (getManager()->isEqual($person)) {

    return “You’re a manager”;

  }

  return null;

}


function getManager(): Person {… some code …}
```

WHY

```
declare(strict_types=1);

function processPerson(Person $person, int $age): ?string {

  if ($age === 18) {

    return "You're 18";

  }

 if (getManager()->isEqual($person)) {

    return "You're a manager";

  }

  return null;

}


function getManager(): Person {… some code …}
```

@daveliddament

WHY

```
declare(strict_types=1);

function processPerson(Person $person, int $age): ?string {

  if ($age === 18) {

    return “You’re 18”;

  }

 if (getManager()->isEqual($person)) {

    return “You’re a manager”;

  }

  return null;

}


function getManager(): Person {… some code …}
```

AGENDA

- What is Static Analysis
- Static Analysis vs Testing
- My story: Journey from no static analysis to advanced tools
    - What is a bug
    - Tools for development and CI
    - Baselining legacy code static analysis results

Dave Liddament

@daveliddament

Lamp Bristol

Organise PHP-SW and Bristol PHP Training

15 years of writing software (C, Java, Python, PHP)

APPROPRIATE APPLICATION OF STATIC ANALYSIS REDUCES THE OVERALL COST OF SOFTWARE DEVELOPMENT.

STATIC ANALYSIS:

STATIC ANALYSIS: IS THIS CORRECT CODE?

```
function process($user) {
  // some implementation
}


$a = 1;
process($a);
```

STATIC ANALYSIS: IS THIS CORRECT CODE?

```
function process($user) {
  // some implementation
}


$a = 1;
process($a);
```

STATIC ANALYSIS: IS THIS CORRECT CODE?

```
function process($user) {
  // some implementation
}


$a = 1;
process($a);
```

STATIC ANALYSIS: IS THIS CORRECT CODE?

```
function process($user) {
   // some implementation
}


$a = 1;
process($a);
```

WHAT ABOUT THIS CODE ?

```
function process(User $user) {
  // some implementation
}


$a = 1;
process($a);
```

WHAT ABOUT THIS CODE ?

```
function process(User $user) {
  // some implementation
}

$a = 1;
process($a);
```

WHAT ABOUT THIS CODE ?

```
function process(User $user) {
  // some implementation
}

$a = 1;
process($a);
```

WHAT ABOUT THIS CODE ?

```
function process(User $user) {
  // some implementation
}


$a = 1;
process($a);
```

Static analysis tells you that your code is incorrect.

TESTING

TESTING

```
function getPrice(string $type): int {
   if ($type === “CHILD") {
       $price = 10;
   }
   if ($type === “ADULT”) {
       $price = 20;
   }
   return $price;
}
```

TESTING

```
function getPrice(string $type): int {
    if ($type === “CHILD") {
        $price = 10;
    }
    if ($type === “ADULT”) {
        $price = 20;
    }
    return $price;
}
```

TESTING

```
function getPrice(string $type): int {
    if ($type === “CHILD") {
        $price = 10;
    }
    if ($type === “ADULT”) {
        $price = 20;
    }
    return $price;
}
```

TESTING

```
function getPrice(string $type): int {
    if ($type === "CHILD") {
        $price = 10;
    }
    if ($type === "ADULT") {
        $price = 20;
    }
    return $price;
}
```

TESTING

```
function getPrice(string $type): int {
    if ($type === "CHILD") {
        $price = 10;
    }
    if ($type === "ADULT") {
        $price = 20;
    }
    return $price;
}
```

TEST CASES

	Input	Expected output
Test 1	CHILD	10
Test 2	ADULT	20

TESTING

```
function getPrice(string $type): int {
    if ($type === “CHILD") {
        $price = 10;
    }
    if ($type === “ADULT”) {
        $price = 20;
    }
    return $price;
}
```

TESTING

```
function getPrice(string $type): int {
    if ($type === “CHILD") {
        $price = 10;
    }
    if ($type === “ADULT”) {
        $price = 20;
    }
    return $price;
}
```

✅ All tests pass

TESTING

```
function getPrice(string $type): int {
   if ($type === “CHILD") {
       $price = 10;
   }
   if ($type === “ADULT”) {
       $price = 20;
   }
   return $price;
}
```

✅ All tests pass

💯 Code coverage

Tests tell you a particular scenario is working correctly.

STATIC ANALYSIS

```
function getPrice(string $type): int {
    if ($type === “CHILD") {
        $price = 10;
    }
    if ($type === “ADULT”) {
        $price = 20;
    }
    return $price;
}
```

STATIC ANALYSIS

```
function getPrice(string $type): int {
    if ($type === “CHILD") {
        $price = 10;
    }
    if ($type === “ADULT”) {
        $price = 20;
    }
    return $price;
}
```

⚠️ Possible undefined variable

STATIC ANALYSIS

```
function getPrice(string $type): int {
    if ($type === "CHILD") {
        $price = 10;
    }
    if ($type === "ADULT") {
        $price = 20;
    }
    return $price;
}
```

⚠️ Possible undefined variable

Static analysis tells you that your code is incorrect.

Tests tell you a particular scenario is working correctly.

Could we test a bit more to remove the need for static analysis?

Could we test a bit more to remove the need for static analysis?

No!

APPROPRIATE APPLICATION OF STATIC ANALYSIS REDUCES THE OVERALL COST OF SOFTWARE DEVELOPMENT.

MY STORY…

MY STORY… CHAPTER 1: CODE LOOKED LIKE THIS…

```
<div class="details-intro">
  <h1>Enter your details</h1>

  <p><img src="<?php echo $assetsPath; ?>image/person.png">
        You're adding details for the following
        team<?php echo (count($team) > 1) ? 's' : ''; ?>
        playing on <strong><?php echo asDate($date); ?>.</strong>
        <br>All fields are required.</p>
```

CHAPTER 1: AND ALSO…

❌ No tests

❌ No invalid syntax highlighting in editor

❌ No automated linting of code

CHAPTER 1: AND ALSO…

❌ No tests

❌ No invalid syntax highlighting in editor

❌ No automated linting of code

CHAPTER 1: AND ALSO…

❌ No tests

❌ No invalid syntax highlighting in editor

❌ No automated linting of code

Real time static analysis

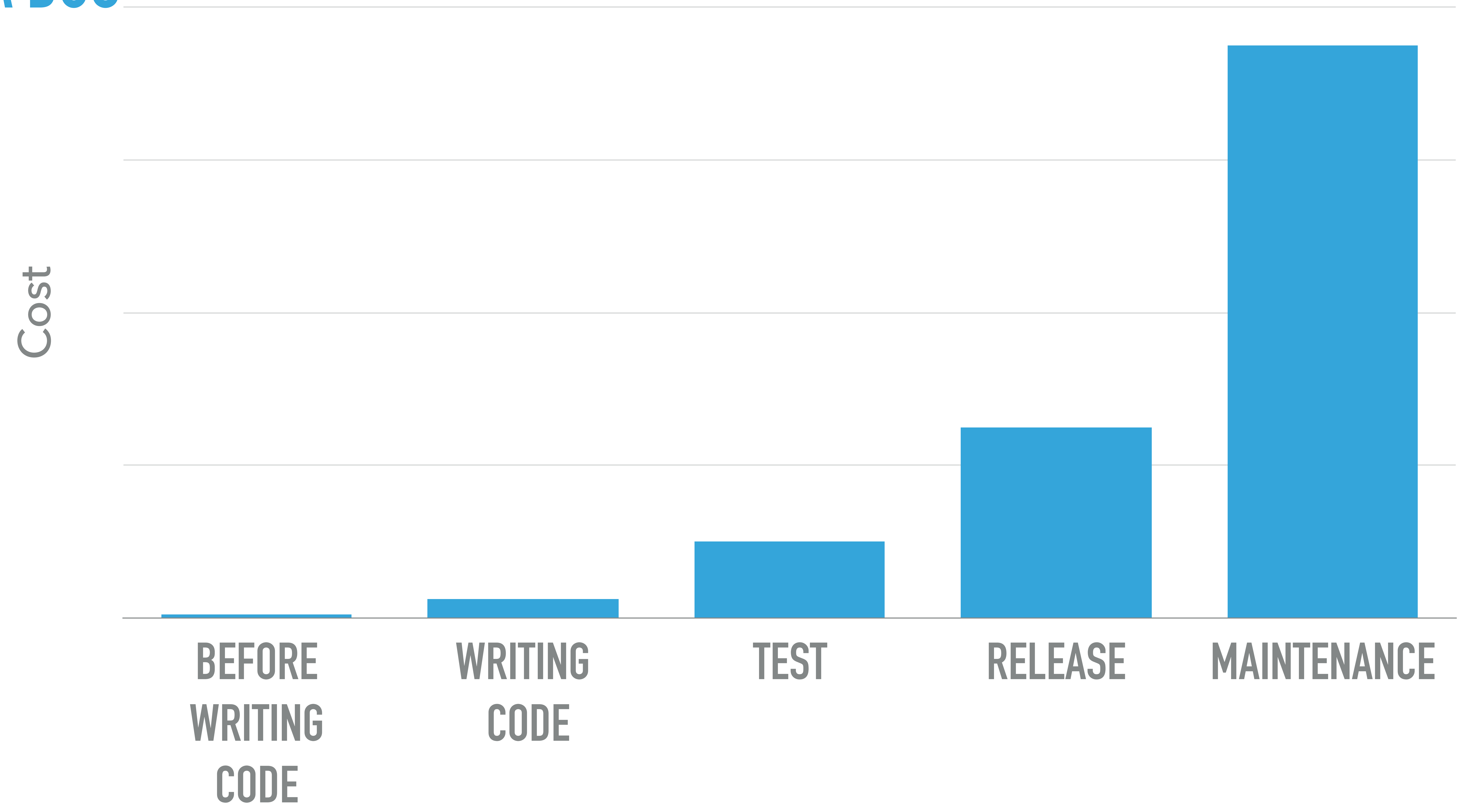
CHAPTER 1: AND ALSO…

❌ No tests

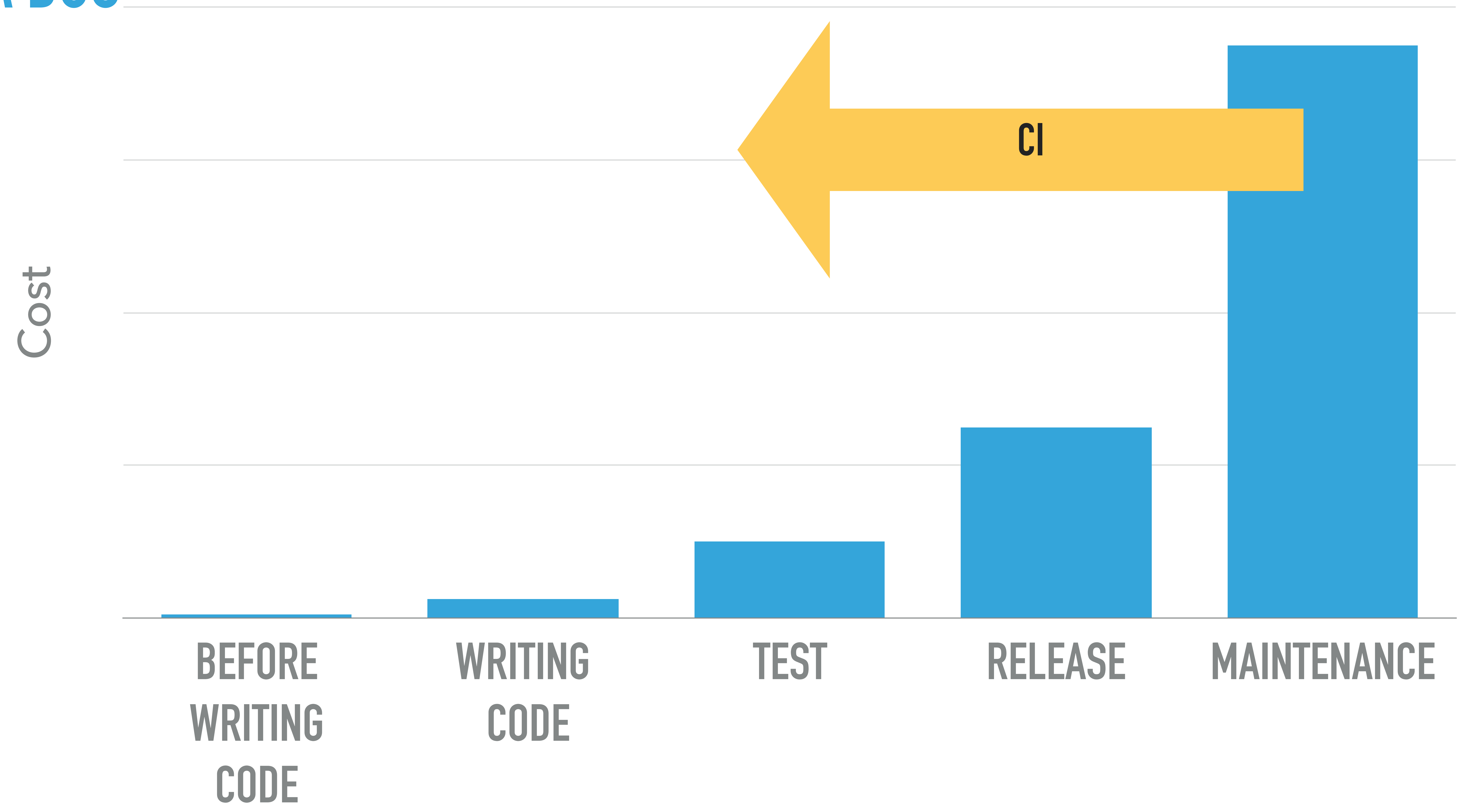
❌ No invalid syntax highlighting in editor — Real time static analysis

❌ No automated linting of code — CI

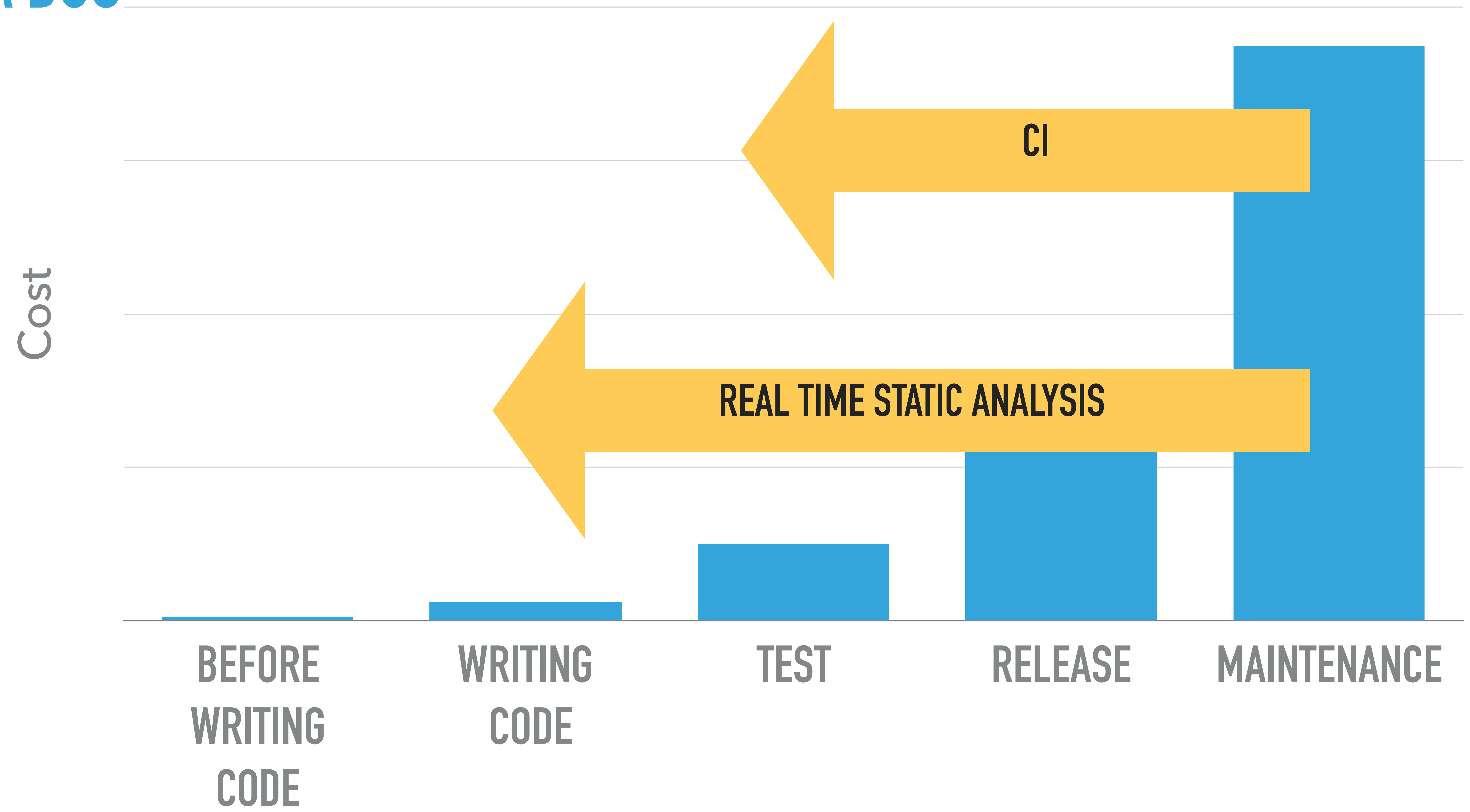
COST OF A BUG


Cost
BEFORE WRITING CODE
WRITING CODE
TEST
RELEASE
MAINTENANCE

COST OF A BUG


Cost
CI
BEFORE WRITING CODE
WRITING CODE
TEST
RELEASE
MAINTENANCE

COST OF A BUG


CI
REAL TIME STATIC ANALYSIS
Cost
BEFORE WRITING CODE
WRITING CODE
TEST
RELEASE
MAINTENANCE

TAKE AWAY: USE A TOOL THAT HIGHLIGHTS SYNTAX ERRORS

```
private function getMarukp(string $markupType, price) {

    if ($markupType === "high") {
        return $price * 10
    }

    retyrn $price;
}
```

TAKE AWAY: USE A TOOL THAT HIGHLIGHTS SYNTAX ERRORS

```
private function getMarukp(string $markupType, price) {

    if ($markupType === "high") {
        return $price * 10
    }

    retyrn $price;
}
```

TAKE AWAY: USE A TOOL THAT HIGHLIGHTS SYNTAX ERRORS

```
private function getMarukp(string $markupType, price) {

    if ($markupType === "high") {
        return $price * 10
    }

    retyrn $price;
}
```

TAKE AWAY: USE A TOOL THAT HIGHLIGHTS SYNTAX ERRORS

```
private function getMarukp(string $markupType, price) {

    if ($markupType === "high") {
        return $price * 10
    }

    retyrn $price;
}
```

TAKE AWAY: USE A TOOL THAT HIGHLIGHTS SYNTAX ERRORS

```
private function getMarukp(string $markupType, price) {

    if ($markupType === "high") {
        return $price * 10
    }

    retyrn $price;
}
```

TAKE AWAY: PERFORM AUTOMATED LINTING AS PART OF CI

- Install:
  - `composer require —dev jakub-onderka/php-parallel-lint`
- Run:
  - `vendor/bin/parallel-lint <directories to scan>`
- E.g.
  - `vendor/bin/parallel-lint src test`

APPROPRIATE APPLICATION OF STATIC ANALYSIS REDUCES THE OVERALL COST OF SOFTWARE DEVELOPMENT.

CHAPTER 2: STATIC ANALYSIS SALESPERSON

CHAPTER 2: STATIC ANALYSIS SALESPERSON

What is a bug?

FOUR TYPES OF 'BUG'

- Bug
- Deferred bug
- Evolvability defect
- False positive

THIS IS A BUG

```
function process(User $user) {
  // some implementation
}

$a = 1;
process($a);
```

THIS IS A BUG

```
function process(User $user) {
  // some implementation
}

$a = 1;
process($a);
```

THIS IS A BUG

```
function process(User $user) {
  // some implementation
}

$a = 1;
process($a);
```

THIS IS A BUG TOO…

```
use Acme\Entity\Person;
function sayHello(Person $person)
{
  echo $person->hi();
}
```

THIS IS A BUG TOO…

```
use Acme\Entity\Person;

function sayHello(Person $person)

{

  echo $person->hi();

}
```

```
namespace Acme\Entity;

class Preson {

  … some code …

}
```

THIS IS A BUG TOO…

```
use Acme\Entity\Person;

function sayHello(Person $person)
{
   echo $person->hi();
}
```

```
namespace Acme\Entity;

class Preson {
   … some code …
}
```

THIS IS A BUG TOO…

```
use Acme\Entity\Person;
function sayHello(Person $person)
{
   echo $person->hi();
}
```

```
namespace Acme\Entity;
class Preson {
   … some code …
}
```

THE GENESIS OF PSALM

Fixing code that ain't broke by Matt Brown

https://medium.com/vimeo-engineering-blog/fixing-code-that-aint-broken-a99e05998c24

Did you find many bugs like this?

Did you find many bugs like this?

Depends on the project

WHAT ABOUT THIS?

WHAT ABOUT THIS?

```
class Person {

/** @var string */
private $name;

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

WHAT ABOUT THIS?

```
class Person {

/** @var string */
private $name;

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

```
$person = new Person();
$person->getName();
```

WHAT ABOUT THIS?

```
class Person {

/** @var string */
private $name;


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

```
$person = new Person();
$person->getName();
```

WHAT ABOUT THIS?

```
class Person {

/** @var string */
private $name;

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

```
$person = new Person();
$person->getName();
```

WHAT ABOUT THIS?

```
class Person {

/** @var string */
private $name;

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

```
$person = new Person();
$person->getName();
```

@daveliddament

THESE ARE DEFERRED BUGS…

```
function getPrice(string $type): int {
    if ($type === “CHILD") {
         $price = 10;
    }
    if ($type === “ADULT”) {
         $price = 20;
    }
    return $price;
}
```

Are “deferred bugs” really bugs?

Are “deferred bugs” really bugs?

Probably quicker to fix than to risk it.

INSTEAD OF?

```
class Person {


/** @var string */
private $name;


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

INSTEAD OF?

```
class Person {

/** @var string */
private $name;

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

USE THIS

```
class Person {

/** @var string */
private $name;

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

Evolvability Defect

CODE THAT MAKES CODE BASE LESS COMPLIANT WITH STANDARDS, MORE ERROR PRONE, OR MORE DIFFICULT TO MODIFY, EXTEND OR UNDERSTAND.

Evolvability Defect

EVOLVABILITY IS IMPORTANT

- Evolvability defects account for 80% of bugs found during code review [1, 2]
- Low evolvability costs money:
    - New features took 28% longer to implement [3]
    - Fixing bugs took 36% longer [3]

AN EVOLVABILITY DEFECT

```
/**
 * @param Person $person
 * @return int
 */
function getAgeNextBirthday($a): string
{
  return "Age next birthday " . $a->asI() + 1;
}
```

AN EVOLVABILITY DEFECT

```
/**
 * @param Person $person
 * @return int
 */
function getAgeNextBirthday($a): string
{
  return "Age next birthday " . $a->asI() + 1;
}
```

AN EVOLVABILITY DEFECT

```
/**
 * @param Person $person
 * @return int
 */
function getAgeNextBirthday($a): string
{
  return “Age next birthday ” . $a->asI() + 1;
}
```

WHAT IS A BUG?

- Bug
- Deferred bug
- Evolvability defect
- False positive

WHAT IS A BUG?

- Bug
- Deferred bug
- Evolvability defect
- False positive

WHAT IS A BUG?

- Bug
- Deferred bug
- Evolvability defect
- False positive

WHAT IS A BUG?

- Bug
- Deferred bug
- Evolvability defect
- False positive

Do you really expect the team to correct 3186 “bugs” before developing new features?

Do you really expect the team to correct 3186 “bugs” before developing new features?

No. Use the baseline.

CHAPTER 3:

CHAPTER 3: JAVA DEVELOPER

Apache maven

Java

checkstyle

eclipse

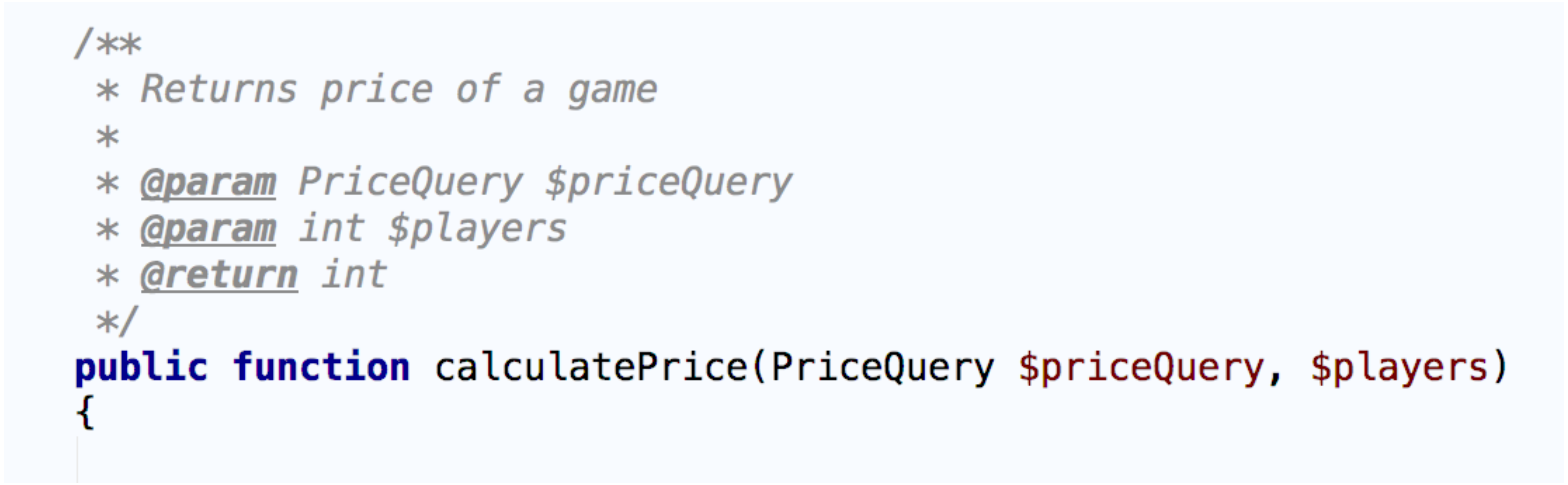
CHAPTER 4: RETURN TO PHP

Apache maven → COMPOSER

eclipse → PS

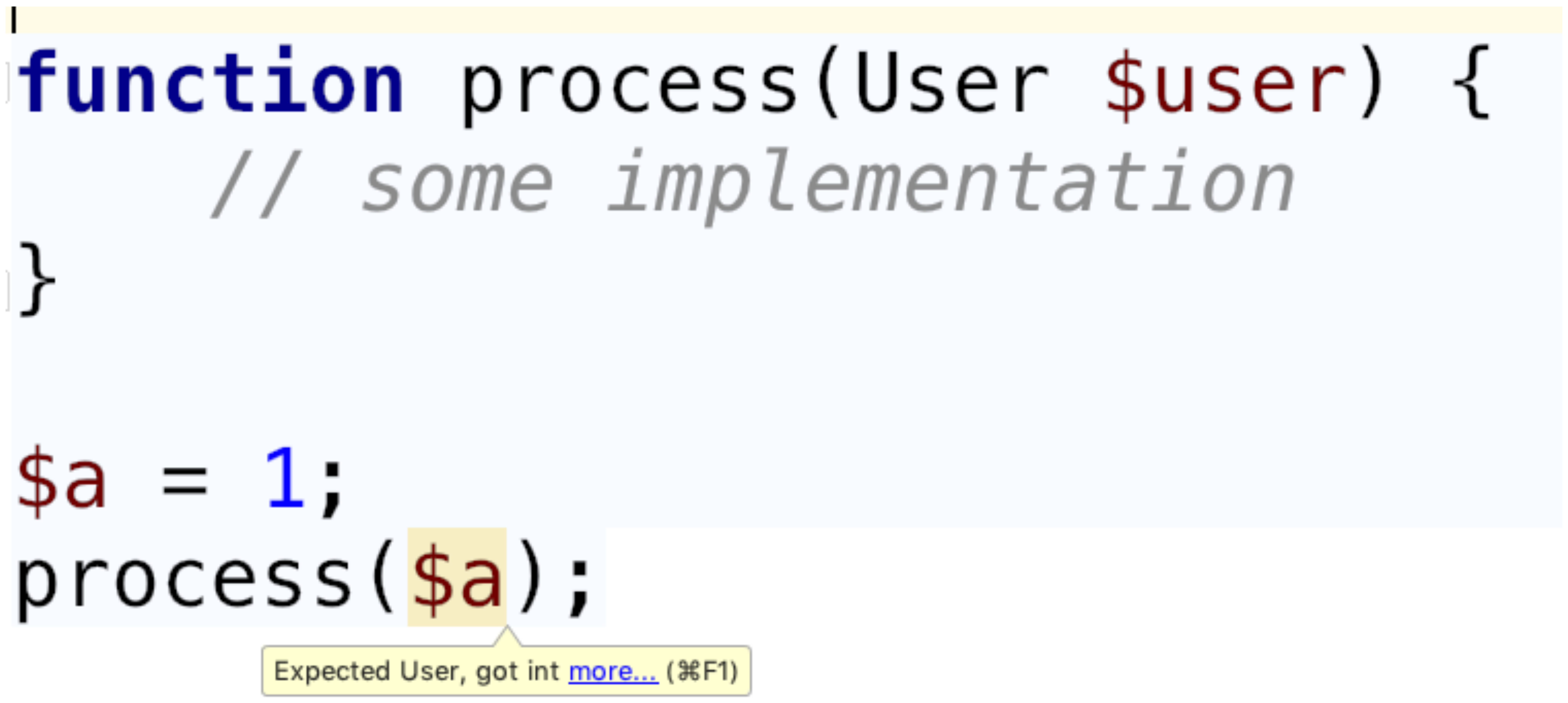
checkstyle → FriendsOfPHP/PHP-CS-Fixer

CHAPTER 4: RETURN TO PHP - TYPE HINT EVERYTHING!



```php
/**
 * Returns price of a game
 *
 * @param PriceQuery $priceQuery
 * @param int $players
 * @return int
 */
public function calculatePrice(PriceQuery $priceQuery, $players)
{
```

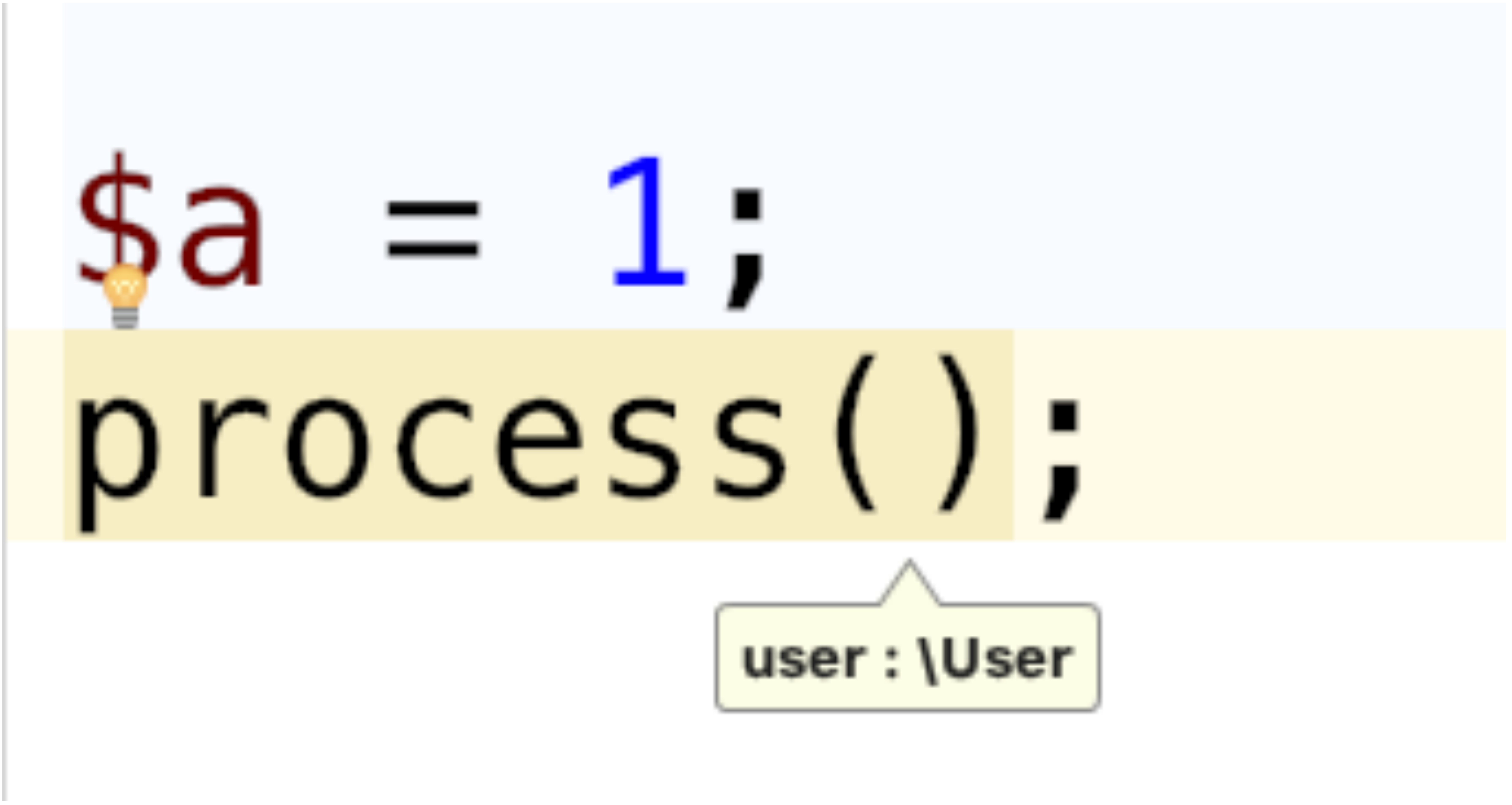
GETTING THE MOST FROM REAL TIME STATIC ANALYSIS



```
function process(User $user) {
    // some implementation
}


$a = 1;
process($a);
```

Expected User, got int more... (⌘F1)

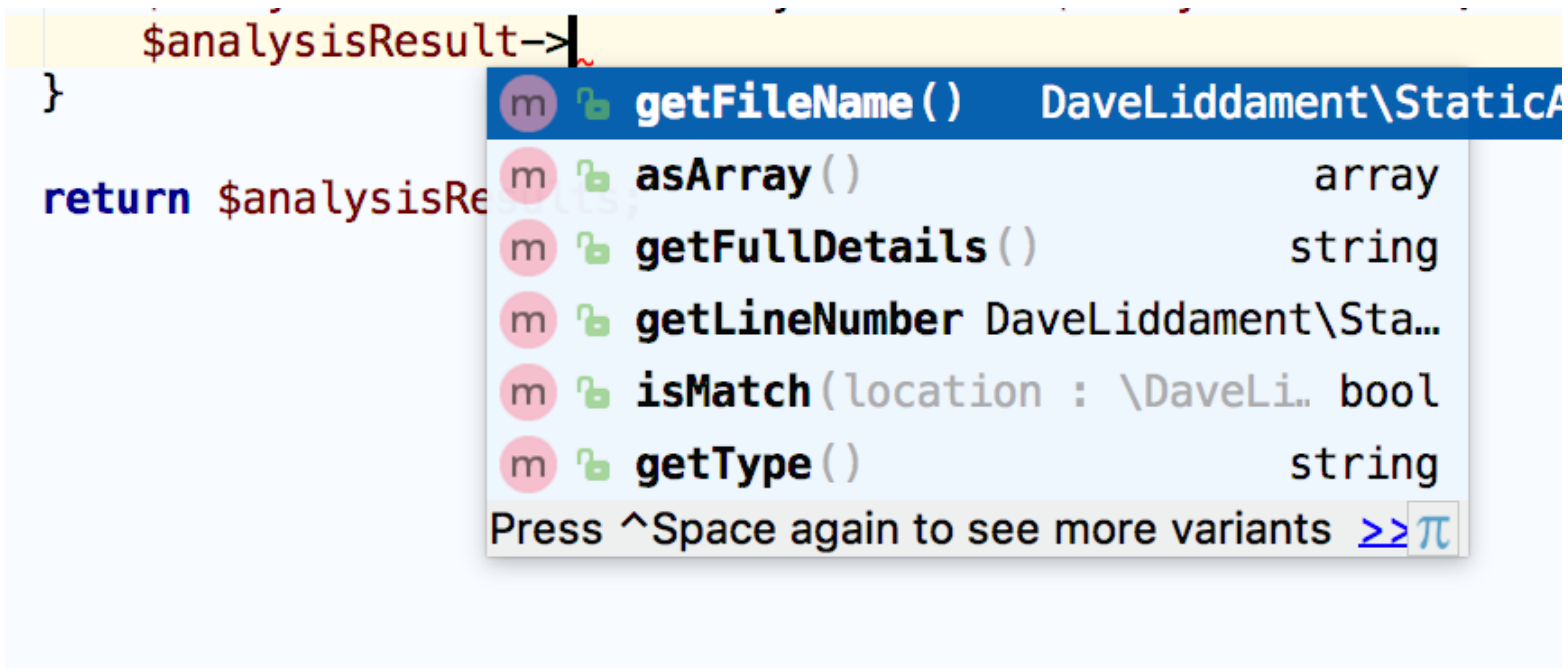
GETTING THE MOST FROM REAL TIME STATIC ANALYSIS



```php
$a = 1;
process();
```

user : \User

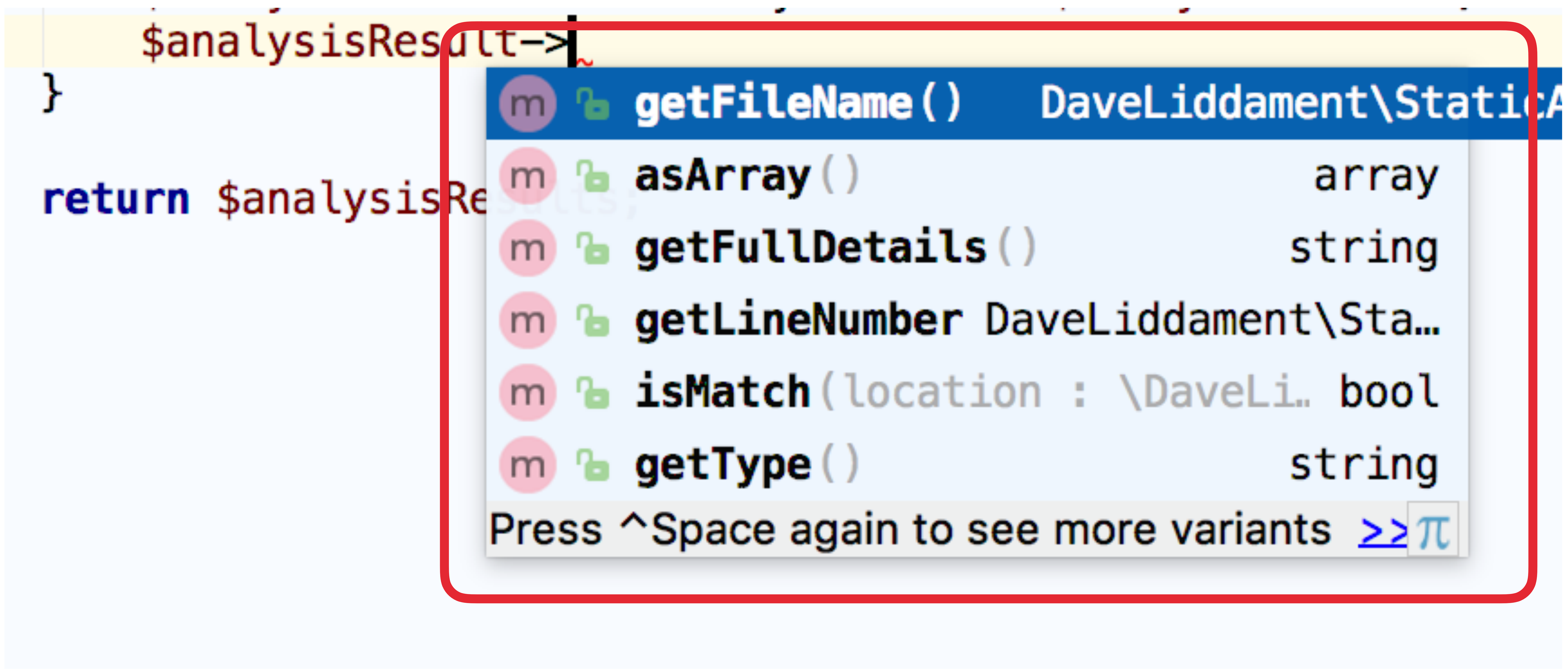
GETTING THE MOST FROM REAL TIME STATIC ANALYSIS



```
    $analysisResult->
}

return $analysisRe
```

```
m getFileName()      DaveLiddament\StaticA
m asArray()                          array
m getFullDetails()                  string
m getLineNumber DaveLiddament\Sta…
m isMatch(location : \DaveLi…        bool
m getType()                         string
Press ^Space again to see more variants >> π
```

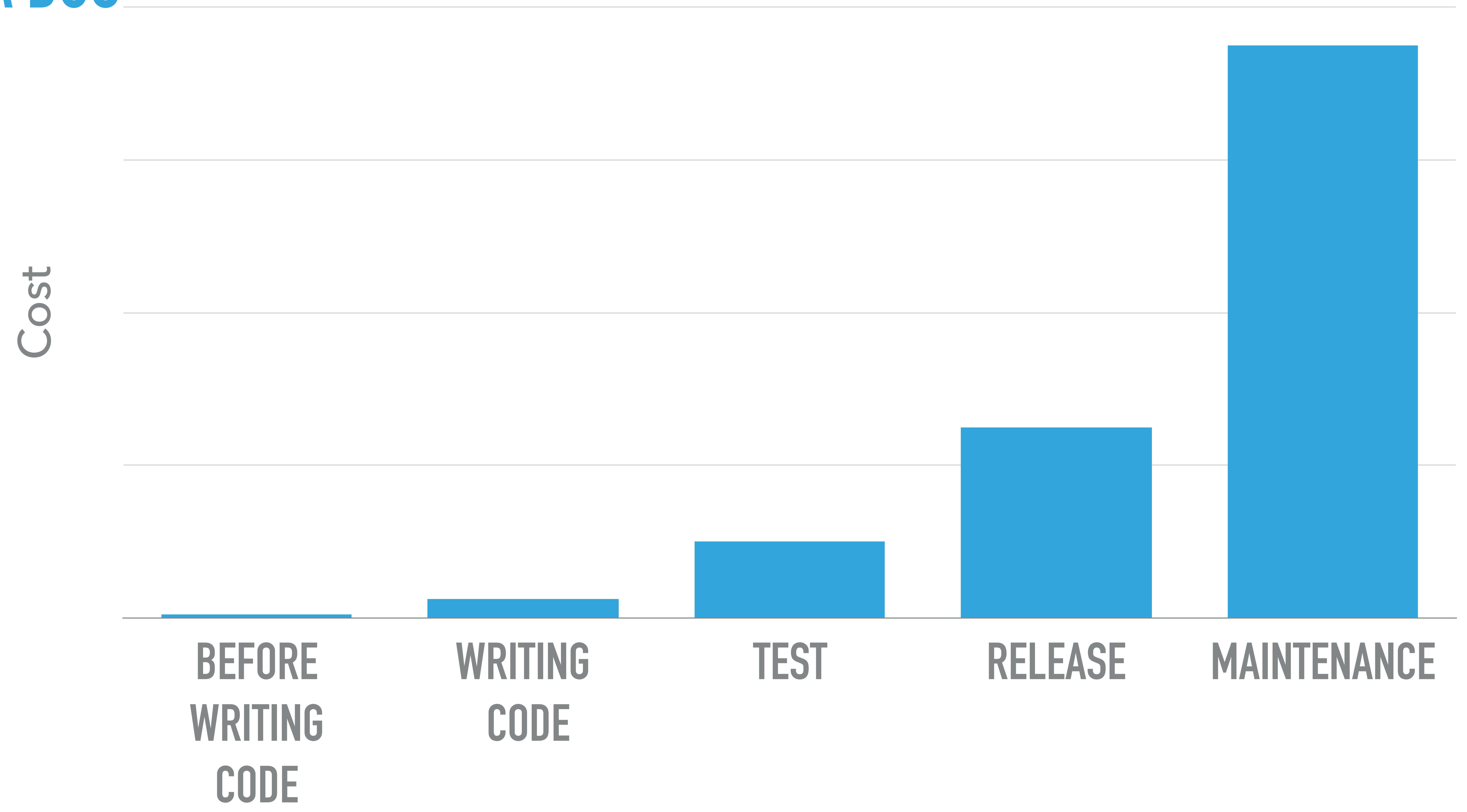
GETTING THE MOST FROM REAL TIME STATIC ANALYSIS



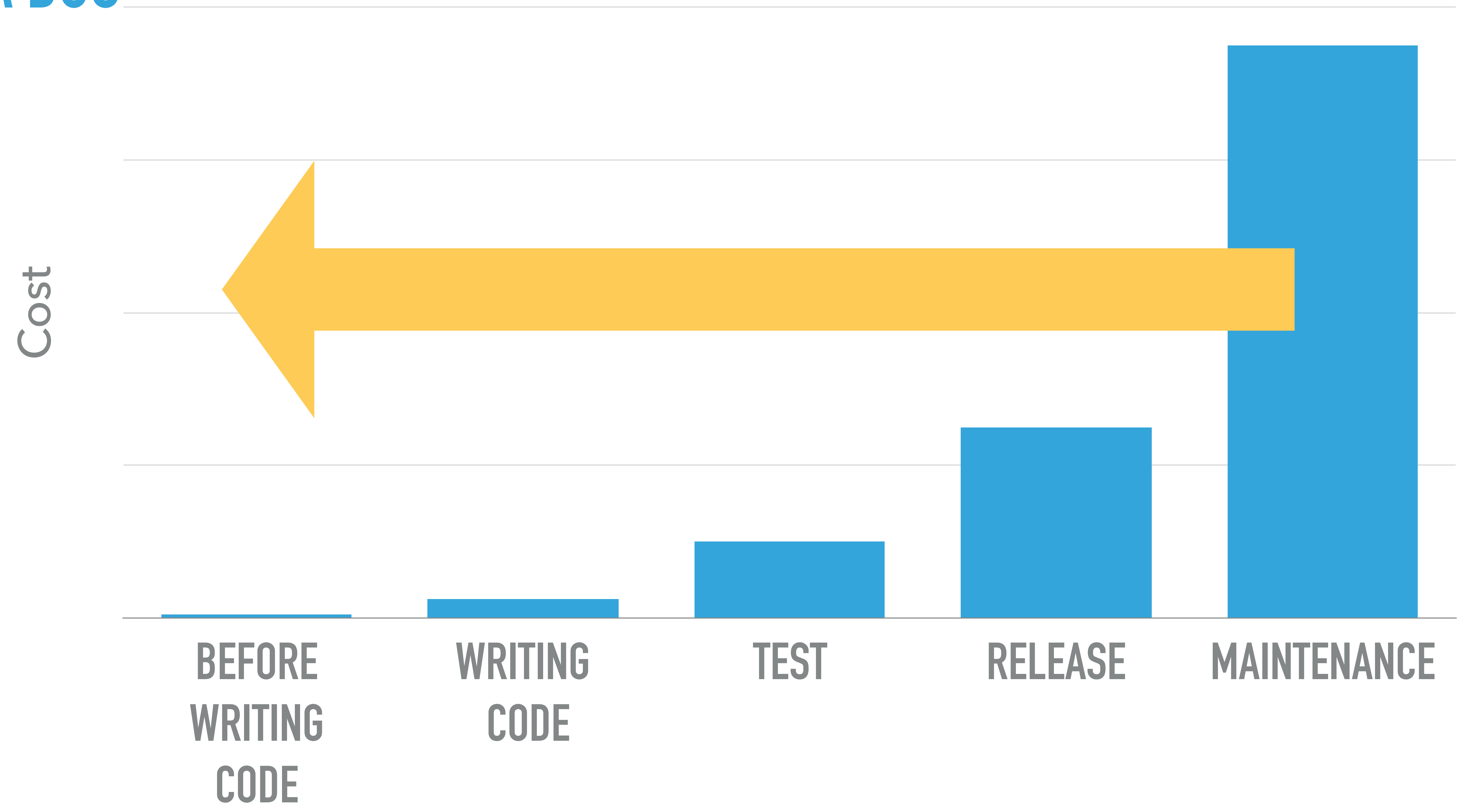
```
    $analysisResult->
}

return $analysisRe
```

```
m  getFileName()               DaveLiddament\StaticA
m  asArray()                                   array
m  getFullDetails()                           string
m  getLineNumber DaveLiddament\Sta…
m  isMatch(location : \DaveLi..                 bool
m  getType()                                  string
Press ^Space again to see more variants >> π
```

COST OF A BUG


Cost
BEFORE WRITING CODE
WRITING CODE
TEST
RELEASE
MAINTENANCE

COST OF A BUG


Cost
BEFORE WRITING CODE
WRITING CODE
TEST
RELEASE
MAINTENANCE

REQUIREMENTS FOR REAL TIME STATIC ANALYSIS TOOL (IDE)

- Understand entire codebase (including vendor directory)
- Highlight errors in real time
- Suggest / autocomplete based on context
- Refactoring (e.g. rename, move, extract)

APPROPRIATE APPLICATION OF STATIC ANALYSIS REDUCES THE OVERALL COST OF SOFTWARE DEVELOPMENT.

CHAPTER 5: HAPPY

Sample PHP/CircleCI project: https://github.com/DaveLiddament/skeleton-ci-project

CI TOOLSET

- Composer validate: **`composer validate --strict`**
- Parallel lint: **`jakub-onderka/php-parallel-lint`**
- PHP CS fixer: **`friendsofsymfony/php-cs-fixer`**
- Var dump checker: **`jakub-onderka/php-var-dump-checker`**
- Security checker: **`sensiolabs/security-checker`**

PHP bible for static analysis tools: https://github.com/exakat/php-static-analysis-tools

CI TOOLSET FOR SYMFONY (3) PROJECTS

- Twig lint: `console lint:twig <dir containing twig templates>`
- Yaml lint: `console lint:yaml <dir containing yaml config>`
- Doctrine : `console doctrine:schema:validate`

STILL THIS NAGGING PROBLEM

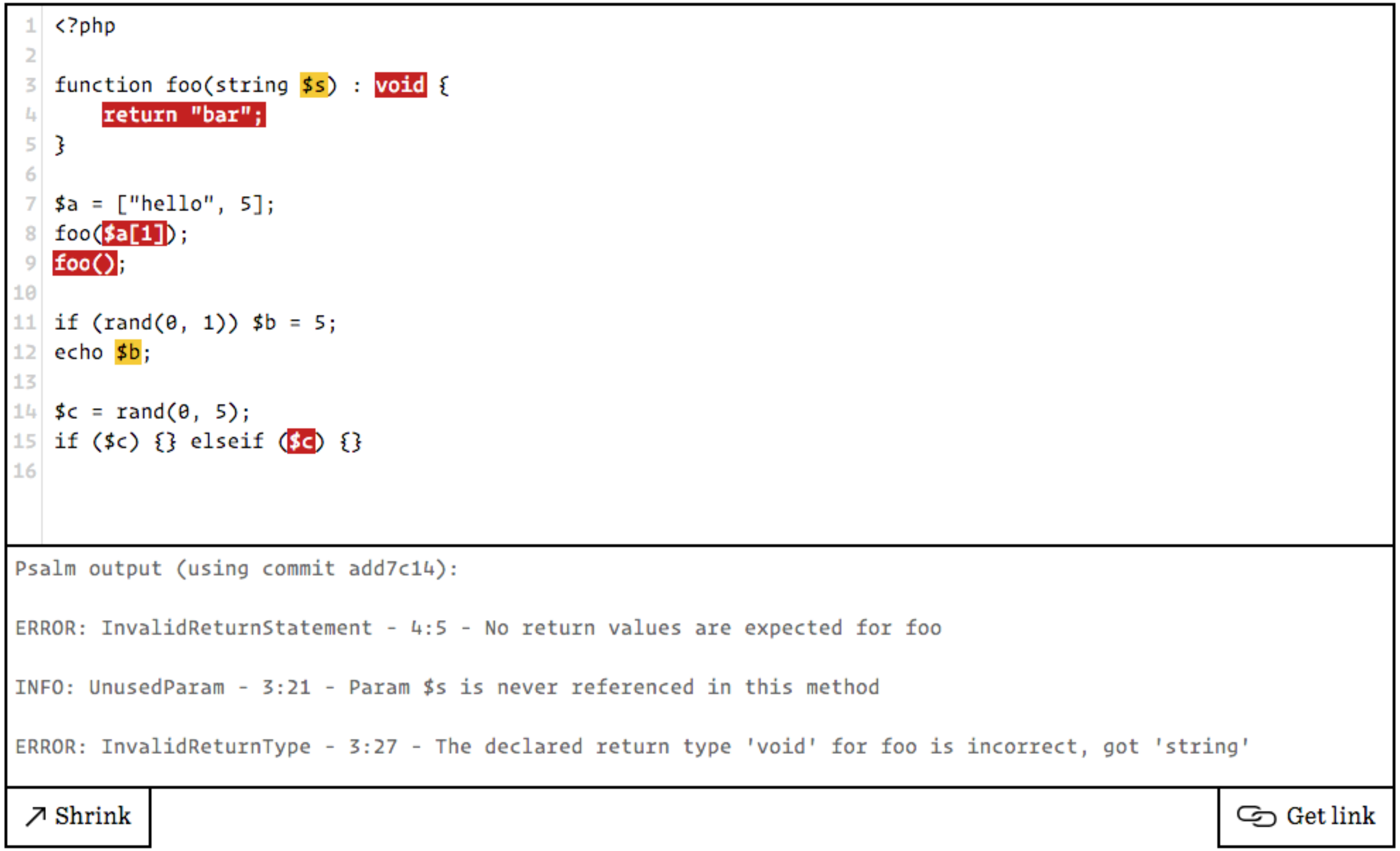
✅ Real time static analysis

❌ CI

CHAPTER 6: ADVANCED STATIC ANALYSIS TOOLS

- Psalm https://getpsalm.org/
- Phan: https://github.com/phan/phan
- PHPStan https://github.com/phpstan/phpstan

ADVANCED STATIC ANALYSIS TOOLS



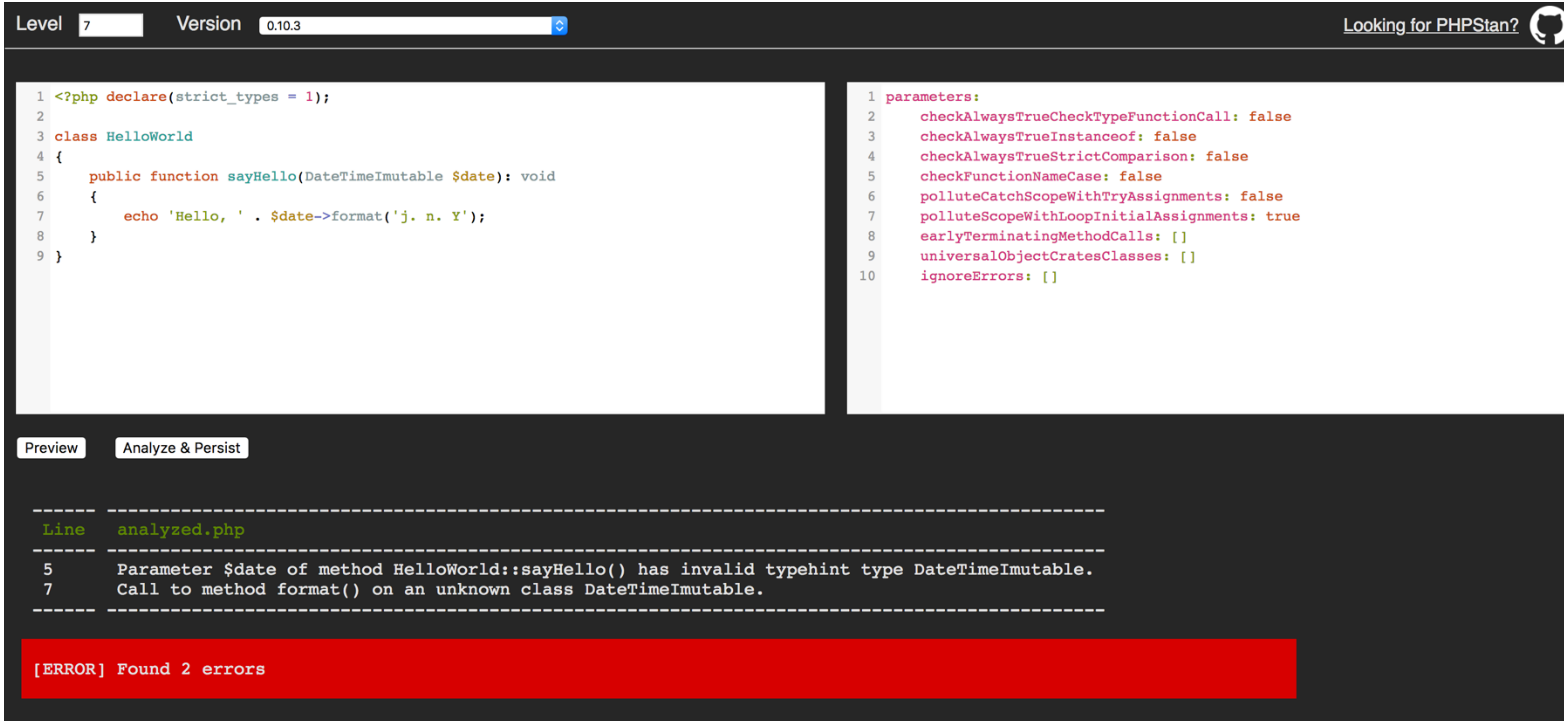
```php
<?php

function foo(string $s) : void {
    return "bar";
}

$a = ["hello", 5];
foo($a[1]);
foo();

if (rand(0, 1)) $b = 5;
echo $b;

$c = rand(0, 5);
if ($c) {} elseif ($c) {}

```

```
Psalm output (using commit add7c14):

ERROR: InvalidReturnStatement - 4:5 - No return values are expected for foo

INFO: UnusedParam - 3:21 - Param $s is never referenced in this method

ERROR: InvalidReturnType - 3:27 - The declared return type 'void' for foo is incorrect, got 'string'
```

https://getpsalm.org

@daveliddament

ADVANCED STATIC ANALYSIS TOOLS



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

class HelloWorld
{
    public function sayHello(DateTimeImutable $date): void
    {
        echo 'Hello, ' . $date->format('j. n. Y');
    }
}
```

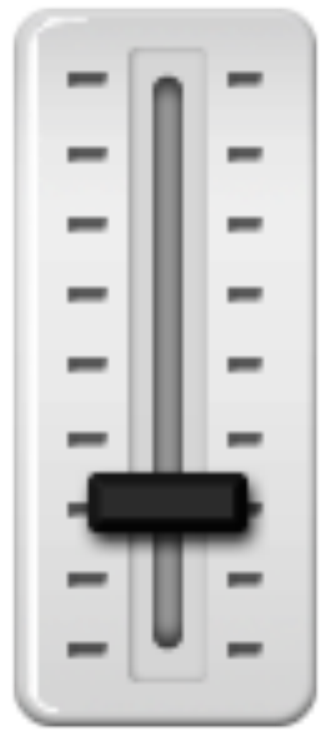
```yaml
parameters:
    checkAlwaysTrueCheckTypeFunctionCall: false
    checkAlwaysTrueInstanceof: false
    checkAlwaysTrueStrictComparison: false
    checkFunctionNameCase: false
    polluteCatchScopeWithTryAssignments: false
    polluteScopeWithLoopInitialAssignments: true
    earlyTerminatingMethodCalls: []
    universalObjectCratesClasses: []
    ignoreErrors: []
```

```
 ------ ---------------------------------------------------------------------------
  Line   analyzed.php
 ------ ---------------------------------------------------------------------------
  5      Parameter $date of method HelloWorld::sayHello() has invalid typehint type DateTimeImutable.
  7      Call to method format() on an unknown class DateTimeImutable.
 ------ ---------------------------------------------------------------------------

 [ERROR] Found 2 errors
```

https://phpstan.org/

@daveliddament

COMMON CONCEPTS: LEVELS



	Least strict	Strictest
Psalm	8	1
Phan	5	1
PHPStan	0	7

COMMON CONCEPTS: GENERICS

```
class Business {


    public function getEmployees(): array {…}

}

function promote(Employee $employee): void {…}




foreach($business->getEmployees() as $employee) {

    promote($employee);


}
```

COMMON CONCEPTS: GENERICS

```
class Business {

    public function getEmployees(): array {…}
}

function promote(Employee $employee): void {…}

foreach($business->getEmployees() as $employee) {
    promote($employee);

}
```

COMMON CONCEPTS: GENERICS

```
class Business {

    public function getEmployees(): array {…}
}

function promote(Employee $employee): void {…}


foreach($business->getEmployees() as $employee) {
    promote($employee);

}
```

COMMON CONCEPTS: GENERICS

```
class Business {

    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}


foreach($business->getEmployees() as $employee) {
    promote($employee);

}
```

COMMON CONCEPTS: GENERICS

```
class Business {


    public function getEmployees(): array {…}

}

function promote(Employee $employee): void {…}



foreach($business->getEmployees() as $employee) {

    promote($employee);


}
```

@daveliddament

COMMON CONCEPTS: GENERICS

```
class Business {

    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}


foreach($business->getEmployees() as $employee) {
    promote($employee);

}
```

COMMON CONCEPTS: GENERICS

```
class Business {


    public function getEmployees(): array {…}

}

function promote(Employee $employee): void {…}




foreach($business->getEmployees() as $employee) {

    promote($employee);


}
```

COMMON CONCEPTS: GENERICS

```
class Business {
    /** @return Employee[] */
    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}


foreach($business->getEmployees() as $employee) {
    promote($employee);

}
```

COMMON CONCEPTS: GENERICS

```
class Business {
    /** @return Employee[] */
    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}



foreach($business->getEmployees() as $employee) {
    promote($employee);

}
```

COMMON CONCEPTS: GENERICS

```
class Business {

    /** @return Employee[] */

    public function getEmployees(): array {…}

}

function promote(Employee $employee): void {…}



foreach($business->getEmployees() as $employee) {

    promote($employee);


}
```

@daveliddament

COMMON CONCEPTS: GENERICS

```
class Business {
    /** @return Employee[] */
    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}



foreach($business->getEmployees() as $employee) {
    promote($employee);

}
```

COMMON CONCEPTS: GENERICS

```
class Business {

    /** @return Employee[] */

    public function getEmployees(): array {…}

}

function promote(Employee $employee): void {…}



foreach($business->getEmployees() as $employee) {

    promote($employee);

}
```

COMMON CONCEPTS: GENERICS

```
class Business {
    /** @return Employee[] */
    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}



foreach($business->getEmployees() as $employee) {
    promote($employee);

}
```

COMMON CONCEPTS: GENERICS

```
class Business {
    /** @return Employee[] */
    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}
function welcome(string $name): void {…}

foreach($business->getEmployees() as $name => $employee) {
    welcome($name);
    promote($employee);
}
```

COMMON CONCEPTS: GENERICS

```
class Business {

    /** @return Employee[] */

    public function getEmployees(): array {…}

}

function promote(Employee $employee): void {…}

function welcome(string $name): void {…}


foreach($business->getEmployees() as $name => $employee) {

    welcome($name);

    promote($employee);

}
```

COMMON CONCEPTS: GENERICS

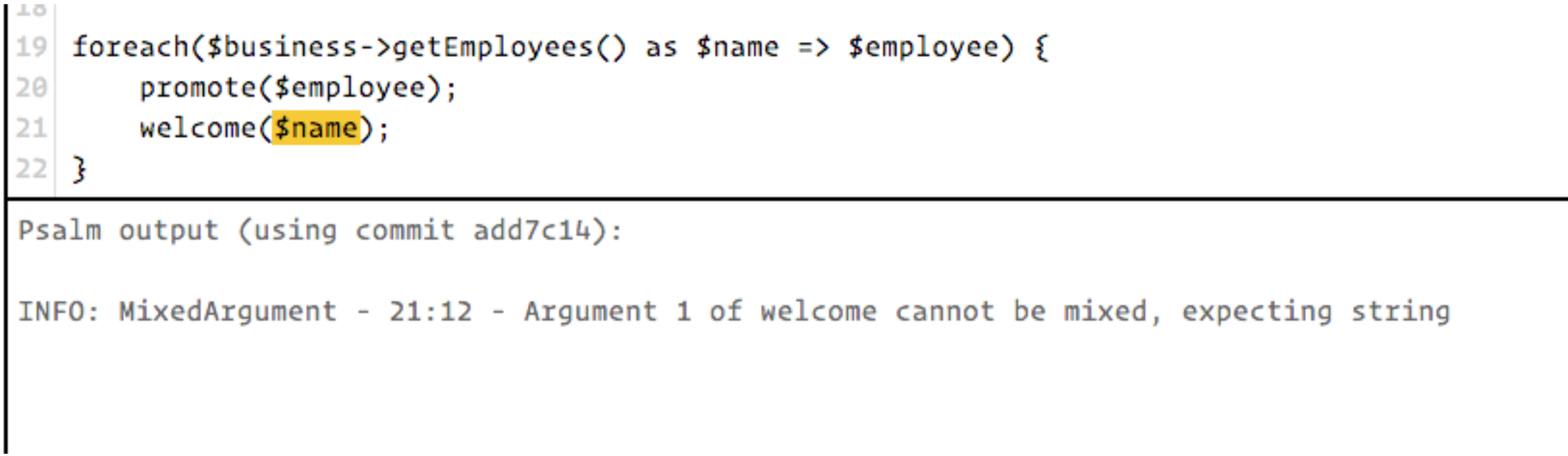
```
class Business {

    /** @return Employee[] */

    public function getEmployees(): array {…}

}

function promote(Employee $employee): void {…}

function welcome(string $name): void {…}


foreach($business->getEmployees() as $name => $employee) {

    welcome($name);

    promote($employee);

}
```

@daveliddament

COMMON CONCEPTS: GENERICS

```
class Business {
    /** @return Employee[] */
    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}
function welcome(string $name): void {…}

foreach($business->getEmployees() as $name => $employee) {
    welcome($name);
    promote($employee);
}
```

COMMON CONCEPTS: GENERICS



```
foreach($business->getEmployees() as $name => $employee) {
    promote($employee);
    welcome($name);
}
```

```
Psalm output (using commit add7c14):

INFO: MixedArgument - 21:12 - Argument 1 of welcome cannot be mixed, expecting string
```

COMMON CONCEPTS: GENERICS

```
class Business {
    /** @return array<string,Employee> */
    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}
function welcome(string $name): void {…}

foreach($business->getEmployees() as $name => $employee) {
    welcome($name);
    promote($employee);
}
```

COMMON CONCEPTS: GENERICS

```
class Business {
    /** @return array<string,Employee> */
    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}
function welcome(string $name): void {…}

foreach($business->getEmployees() as $name => $employee) {
    welcome($name);
    promote($employee);
}
```

COMMON CONCEPTS: GENERICS

```
class Business {
    /** @return array<string,Employee> */
    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}
function welcome(string $name): void {…}

foreach($business->getEmployees() as $name => $employee) {
    welcome($name);
    promote($employee);
}
```

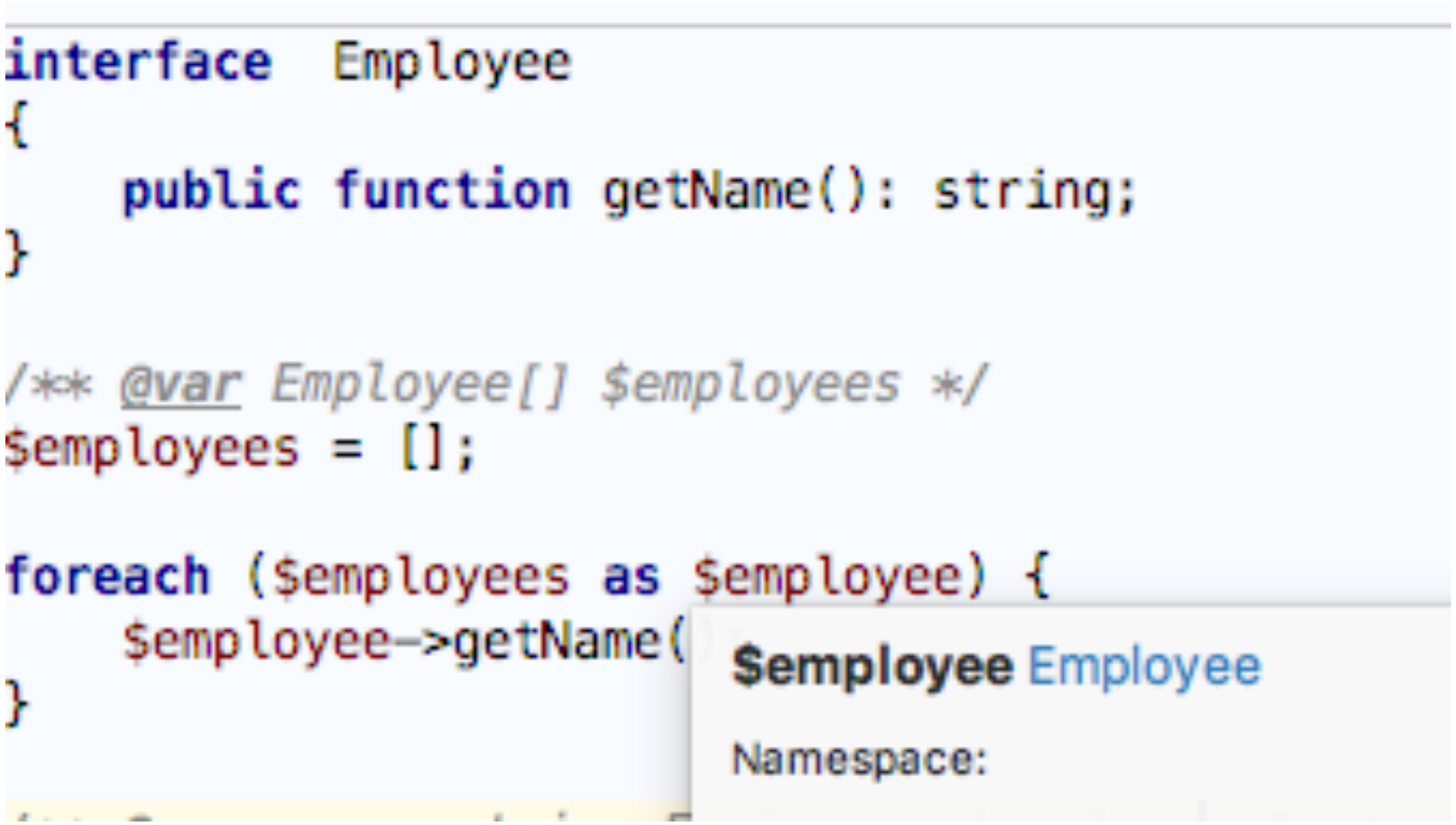
COMMON CONCEPTS: GENERICS

```
class Business {
    /** @return array<string,Employee> */
    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}
function welcome(string $name): void {…}

foreach($business->getEmployees() as $name => $employee) {
    welcome($name);
    promote($employee);
}
```

COMMON CONCEPTS: GENERICS

```
class Business {
    /** @return array<string,Employee> */
    public function getEmployees(): array {…}
}
function promote(Employee $employee): void {…}
function welcome(string $name): void {…}

foreach($business->getEmployees() as $name => $employee) {
    welcome($name);
    promote($employee);
}
```

COMMON CONCEPTS: GENERICS

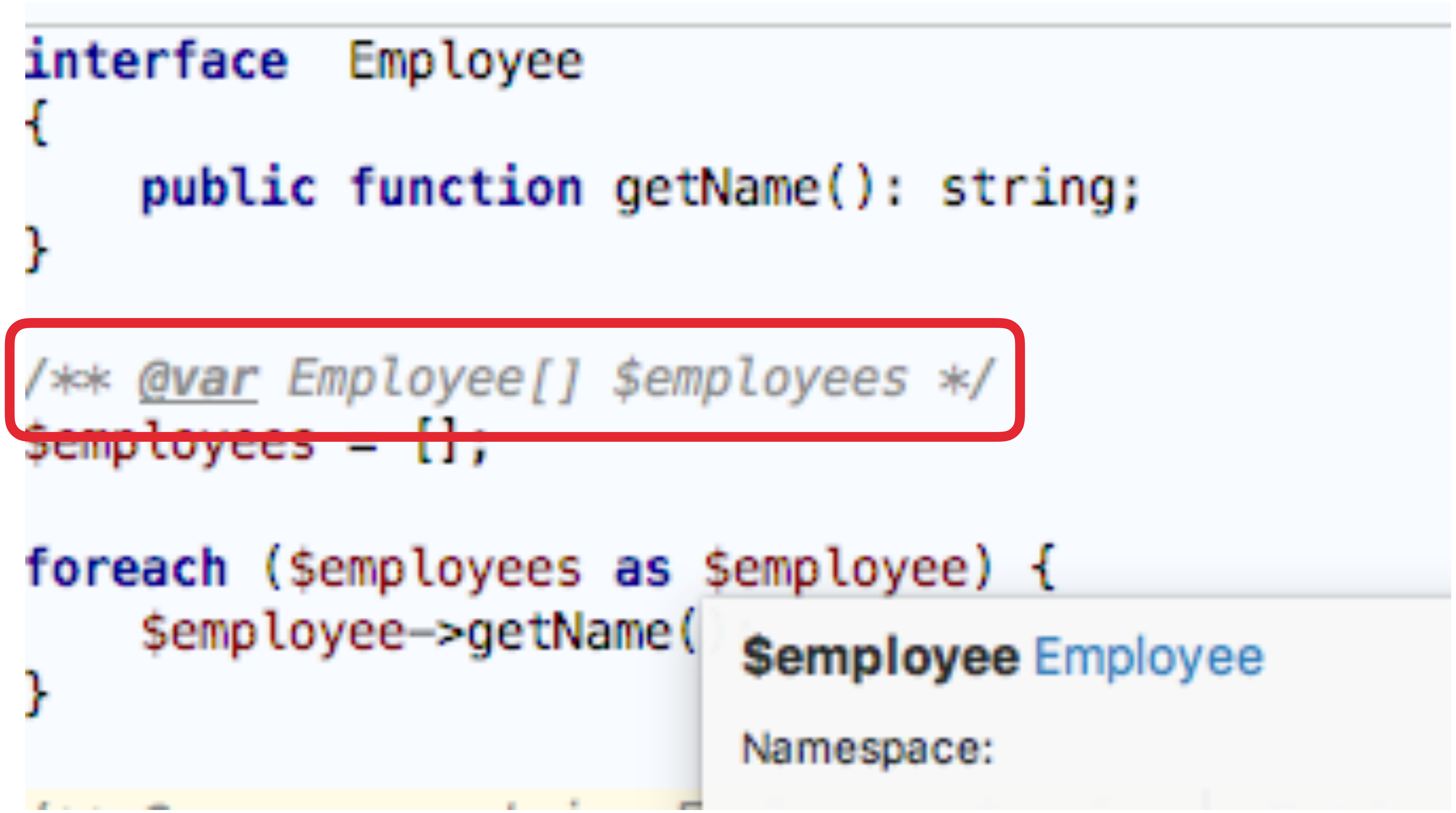


```
interface  Employee
{
    public function getName(): string;
}

/** @var Employee[] $employees */
$employees = [];

foreach ($employees as $employee) {
    $employee->getName(
}
```

$employee Employee

Namespace:

COMMON CONCEPTS: GENERICS

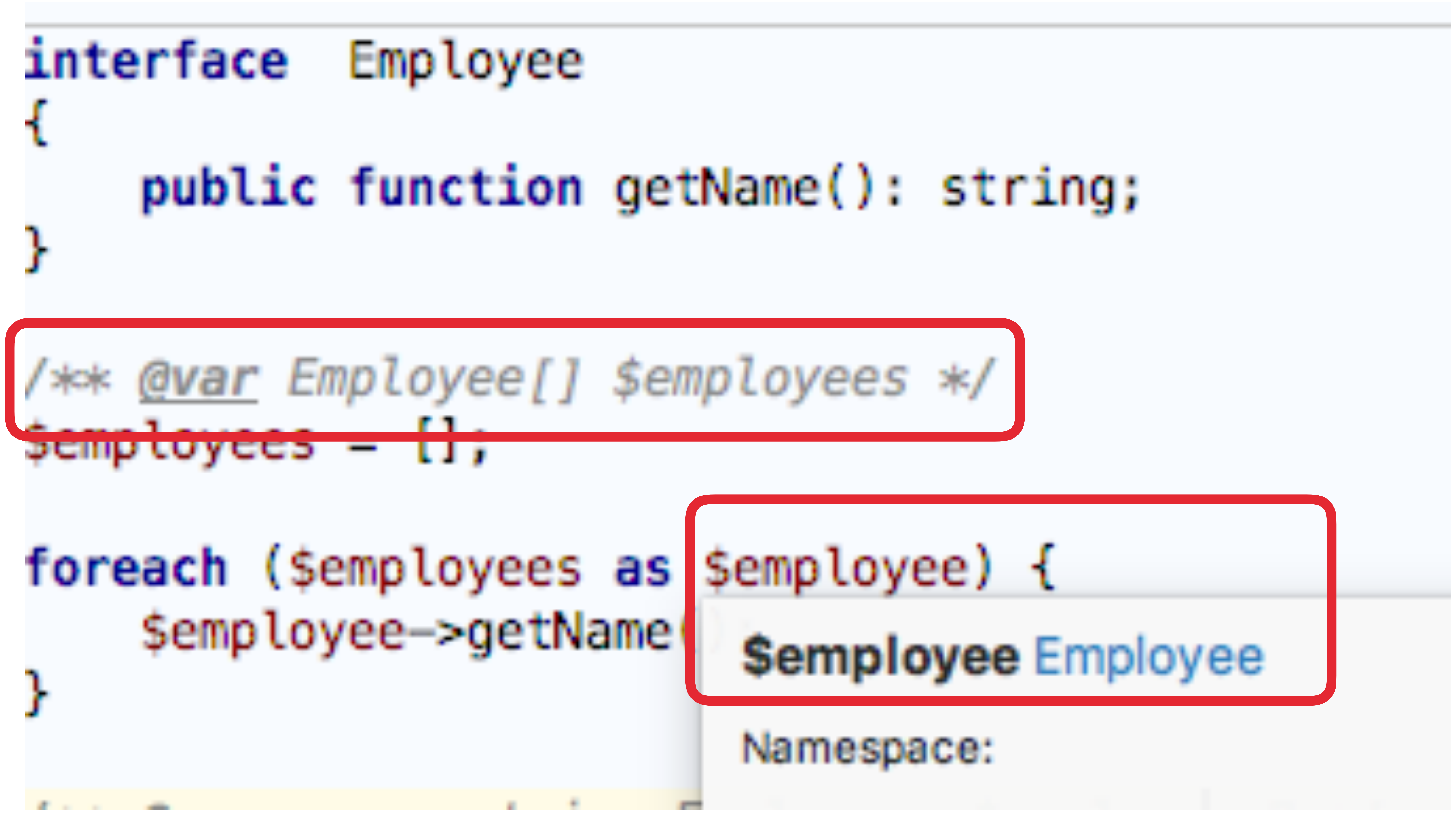


```
interface  Employee
{
    public function getName(): string;
}

/** @var Employee[] $employees */
$employees = [];

foreach ($employees as $employee) {
    $employee->getName(
}
```

$employee Employee

Namespace:

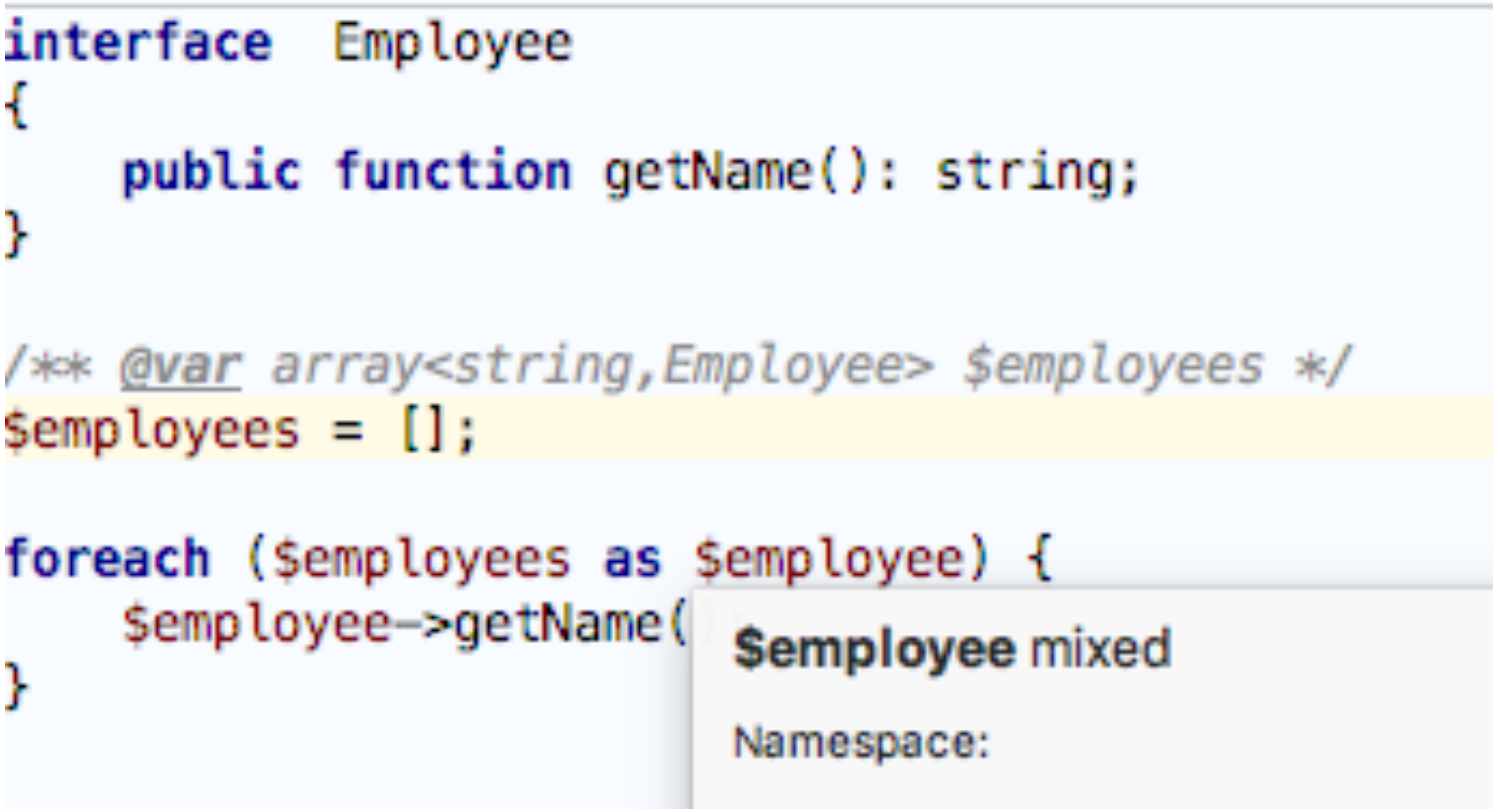
COMMON CONCEPTS: GENERICS



```
interface  Employee
{
    public function getName(): string;
}

/** @var Employee[] $employees */
$employees = [];

foreach ($employees as $employee) {
    $employee->getName(
}
```

```
$employee Employee

Namespace:
```

COMMON CONCEPTS: GENERICS

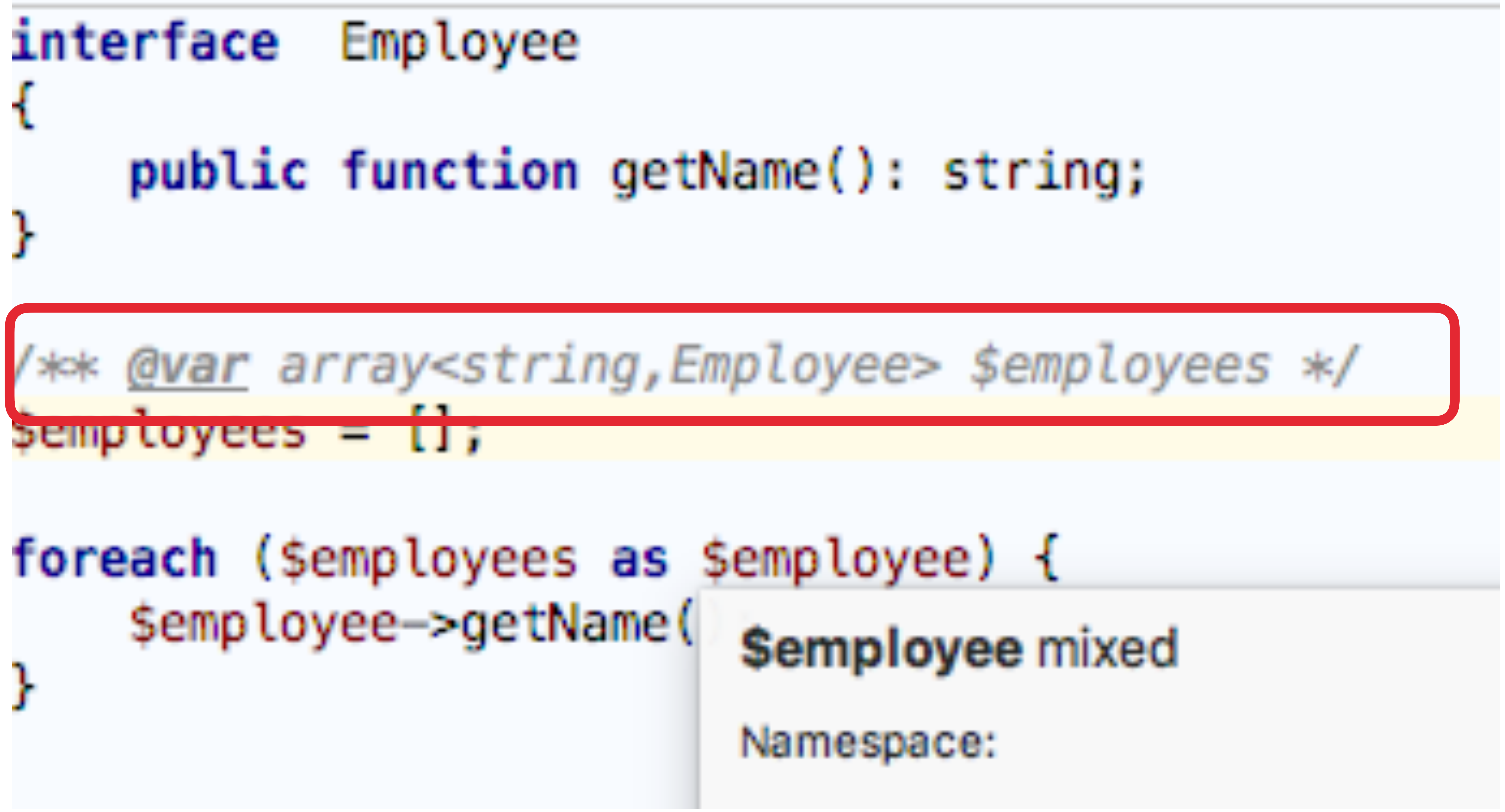


```
interface  Employee
{
    public function getName(): string;
}

/** @var array<string,Employee> $employees */
$employees = [];

foreach ($employees as $employee) {
    $employee->getName(
}
```

$employee mixed

Namespace:

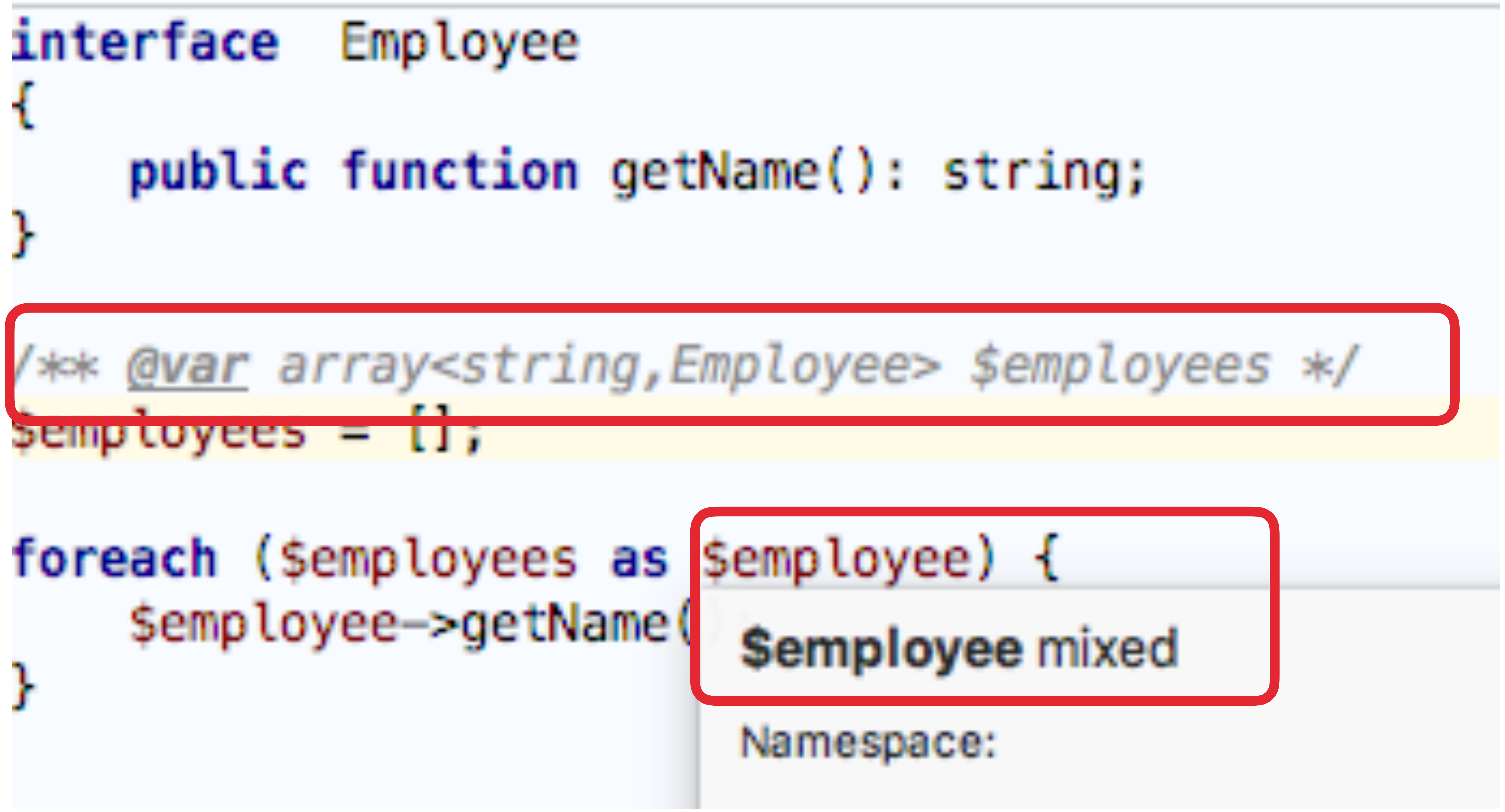
COMMON CONCEPTS: GENERICS



```
interface  Employee
{
    public function getName(): string;
}

/** @var array<string,Employee> $employees */
$employees = [];

foreach ($employees as $employee) {
    $employee->getName(
}
```

```
$employee mixed
Namespace:
```

COMMON CONCEPTS: GENERICS



```
interface  Employee
{
    public function getName(): string;
}

/** @var array<string,Employee> $employees */
$employees = [];

foreach ($employees as $employee) {
    $employee->getName(
}
```

$employee mixed

Namespace:

@daveliddament

COMMON CONCEPTS: GENERICS

```
class Business {
    /**
     * @return Employee[]
     * @psalm-return array<string,Employee>
     */
    public function getEmployees(): array {…}
}
```

COMMON CONCEPTS: GENERICS

```
class Business {
    /**
     * @return Employee[]
     * @psalm-return array<string,Employee>
     */
    public function getEmployees(): array {…}
}
```

COMMON CONCEPTS: GENERICS

```
class Business {
    /**
     * @return Employee[]
     * @psalm-return array<string,Employee>
     */
    public function getEmployees(): array {…}
}
```

COMMON CONCEPTS: GENERICS

```
class Business {
    /**
     * @return Employee[]
     * @psalm-return array<string,Employee>
     */
    public function getEmployees(): array {…}
}
```

PSR-5: PHPDoc: https://github.com/php-fig/fig-standards/blob/master/proposed/phpdoc.md

COMMON CONCEPTS: GENERICS

- In addition to normal annotations:
  - `@var, @param, @return`
- In Psalm:
  - `@psalm-var, @psalm-param, @psalm-return`
- In Phan:
  - `@phan-var, @phan-param, @phan-return`

COMMON CONCEPTS: IGNORE VIOLATIONS

- Set level
- Annotate code:
  - `@psalm-suppress <Issue>`
- Config:
  - Ignore directory
  - Turn off errors
  - Ignore types of errors in certain directories

PSALM: GETTING STARTED

PSALM: GETTING STARTED

- Install:
  - `composer require —-dev vimeo/psalm`

PSALM: GETTING STARTED

- Install:
  - `composer require —-dev vimeo/psalm`
- Create config file:
  - `vendor/bin/psalm —init <directory> <level>`

PSALM: GETTING STARTED

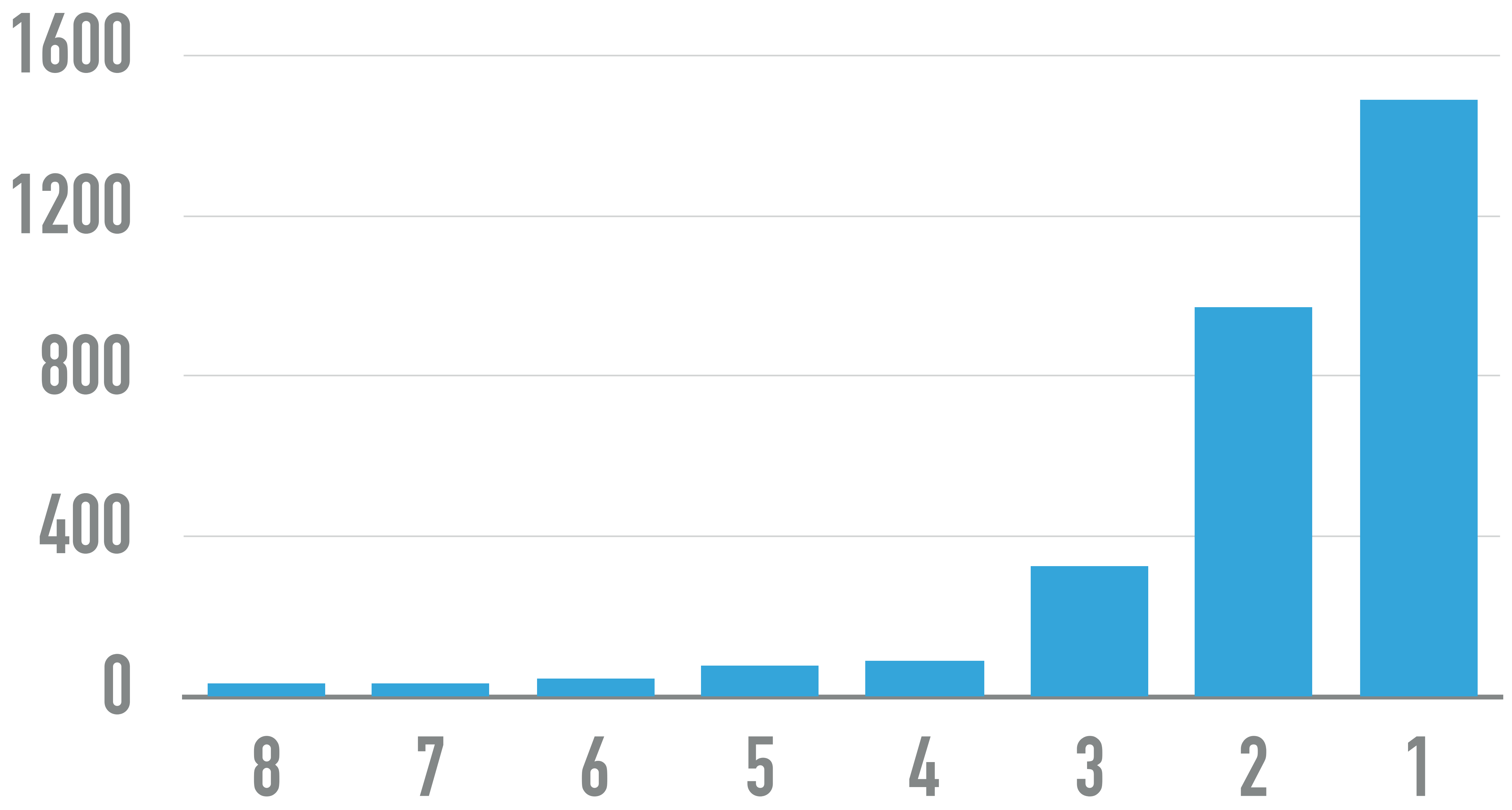
- Install:
  - `composer require —-dev vimeo/psalm`
- Create config file:
  - `vendor/bin/psalm —init <directory> <level>`
- Run:
  - `vendor/bin/psalm`

PSALM: GETTING STARTED

- Install:
  - `composer require —-dev vimeo/psalm`
- Create config file:
  - `vendor/bin/psalm —init <directory> <level>`
- Run:
  - `vendor/bin/psalm`
- `Cry.`

RESULTS

RESULTS



A REAL BUG

```
private function getEmailAddress(array $row): string
{
    $email = $row[self::EMAIL];
    if (empty($email)) {
        throw new ImportEntryException('Invalid or missing email address');
    }

    return $email;
}
```

A REAL BUG

```
private function getEmailAddress(array $row): string
{
    $email = $row[self::EMAIL];
    if (empty($email)) {
        throw new ImportEntryException('Invalid or missing email address');
    }

    return $email;
}
```

A REAL BUG

```
private function getEmailAddress(array $row): string
{
    $email = $row[self::EMAIL];
    if (empty($email)) {
        throw new ImportEntryException('Invalid or missing email address');
    }

    return $email;
}
```

A REAL BUG

```
private function getEmailAddress(array $row): string
{
    $email = $row[self::EMAIL];
    if (empty($email)) {
        throw new ImportEntryException('Invalid or missing email address');
    }

    return $email;
}
```

A REAL BUG

```
private function getEmailAddress(array $row): string
{
    $email = $row[self::EMAIL];
    if (empty($email)) {
        throw new ImportEntryException('Invalid or missing email address');
    }

    return $email;
}
```

A DEFERRED BUG

```
class Location {
  public function getSlug(): ?string {…}
}

function createSearchTerm(Postcode $postcode, string $slug): SearchTerm {…}


… some code …

$searchTerm = createSearchTerm($postcode, $location->getSlug());
```

A DEFERRED BUG

```
class Location {
  public function getSlug(): ?string {…}
}

function createSearchTerm(Postcode $postcode, string $slug): SearchTerm {…}


… some code …

$searchTerm = createSearchTerm($postcode, $location->getSlug());
```

A DEFERRED BUG

```
class Location {
  public function getSlug(): ?string {…}
}

function createSearchTerm(Postcode $postcode, string $slug): SearchTerm {…}


… some code …

$searchTerm = createSearchTerm($postcode, $location->getSlug());
```

A DEFERRED BUG

```
class Location {
  public function getSlug(): ?string {…}
}

function createSearchTerm(Postcode $postcode, string $slug): SearchTerm {…}


… some code …

$searchTerm = createSearchTerm($postcode, $location->getSlug());
```

A DEFERRED BUG

```
class Location {
  public function getSlug(): ?string {…}
}

function createSearchTerm(Postcode $postcode, string $slug): SearchTerm {…}


… some code …

$searchTerm = createSearchTerm($postcode, $location->getSlug());
```

A DEFERRED BUG

```
class Location {
  public function getSlug(): ?string {…}
}

function createSearchTerm(Postcode $postcode, string $slug): SearchTerm {…}


… some code …

$searchTerm = createSearchTerm($postcode, $location->getSlug());
```

A DEFERRED BUG

```
/**
 * @return Location[]|null
 */
function getLocations: ?array {…}

foreach(getLocations() as $location) {
  …
}
```

A DEFERRED BUG

```
/**
 * @return Location[]|null
 */
function getLocations: ?array {…}

foreach(getLocations() as $location) {
  …
}
```

A DEFERRED BUG

```
/**
 * @return Location[]|null
 */
function getLocations: ?array {…}

foreach(getLocations() as $location) {
  …
}
```

EVOLVABILITY DEFECT

```
$plots = array_map(function(Bookmark $bookmark)        {
      return $bookmark->getPlot();
   },$bookmarks);
```

EVOLVABILITY DEFECT

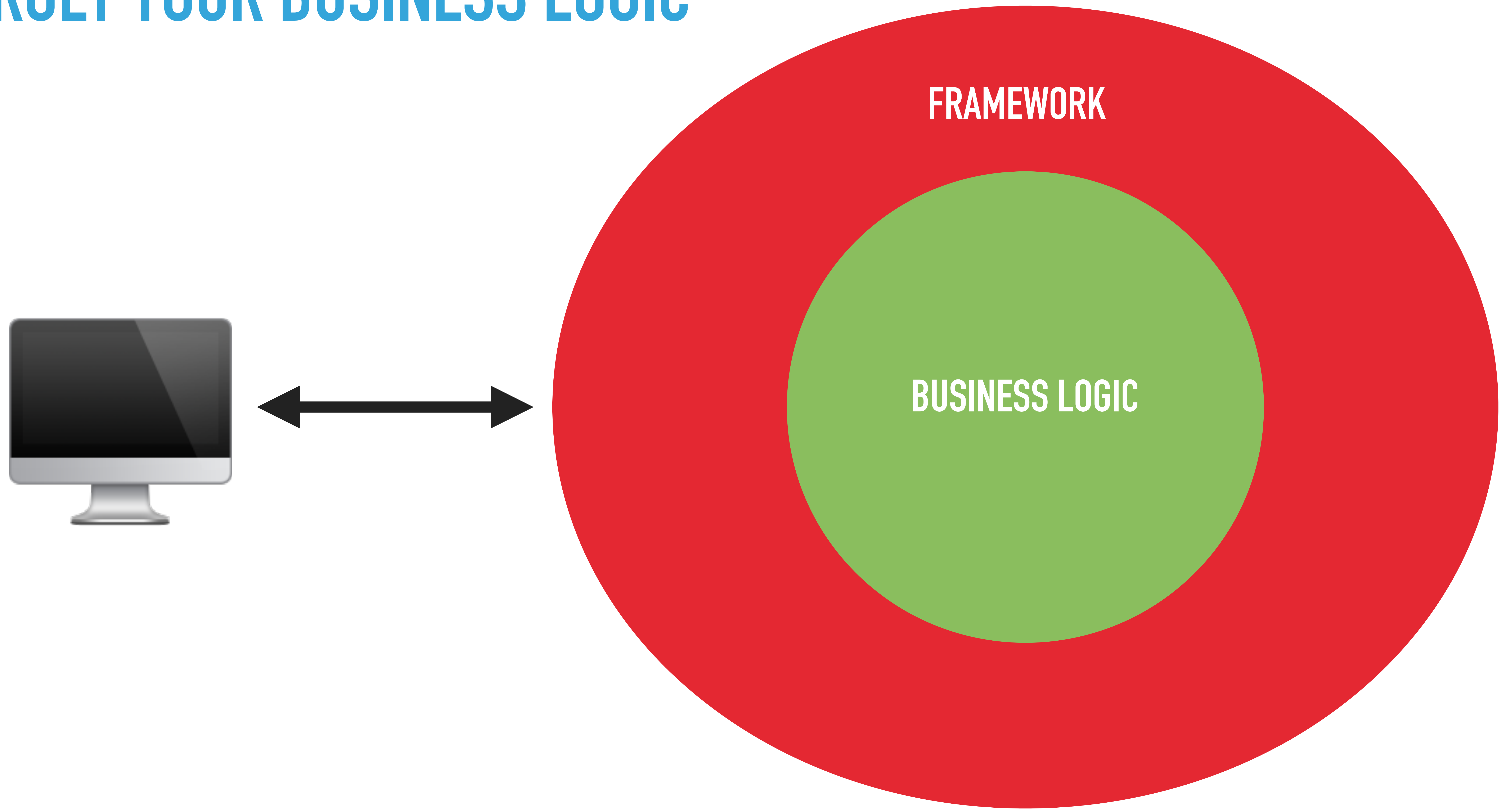
```
$plots = array_map(function(Bookmark $bookmark) :Plot  {
      return $bookmark->getPlot();
   },$bookmarks);
```

You don’t really expect me to fix all those “bugs”?

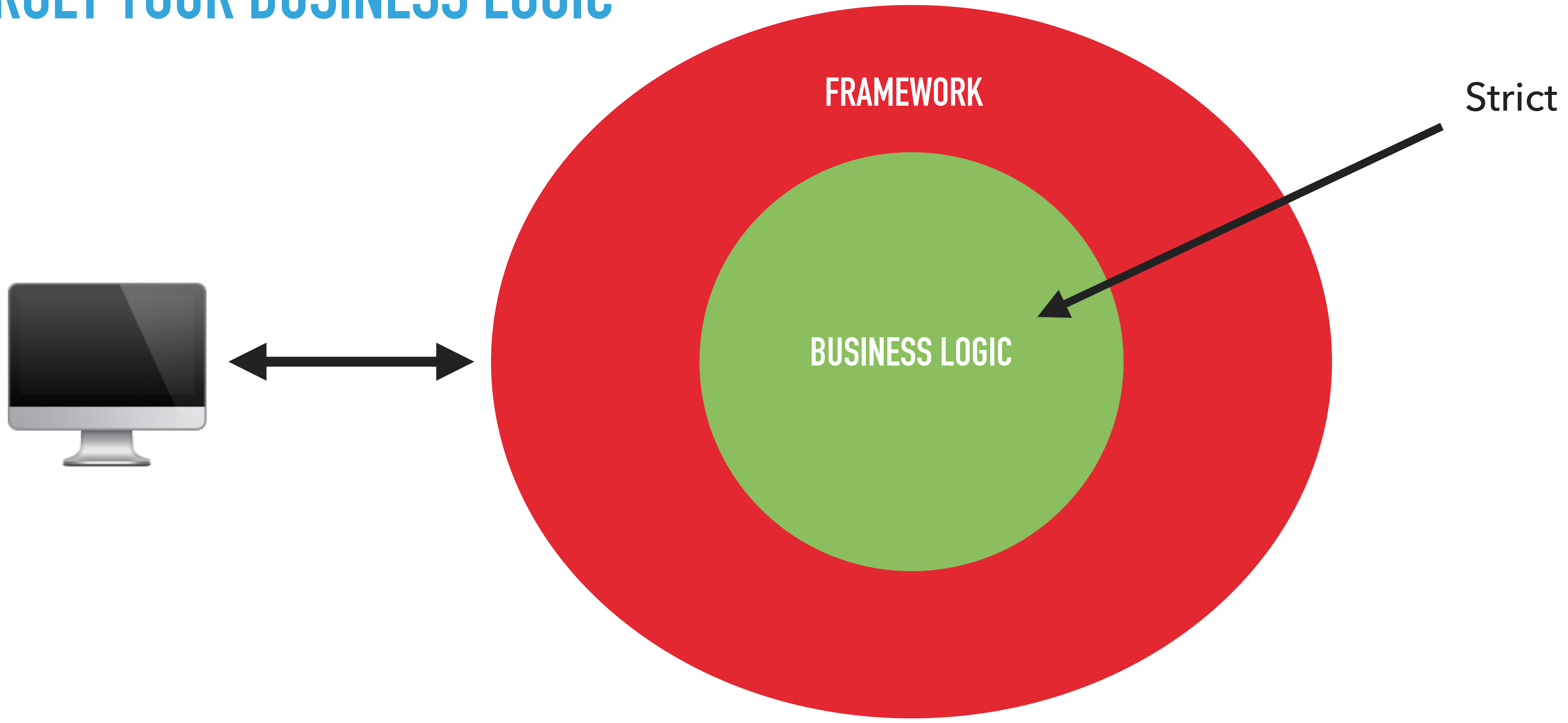
You don’t really expect me to fix all those “bugs”?

No. Here are some tips.

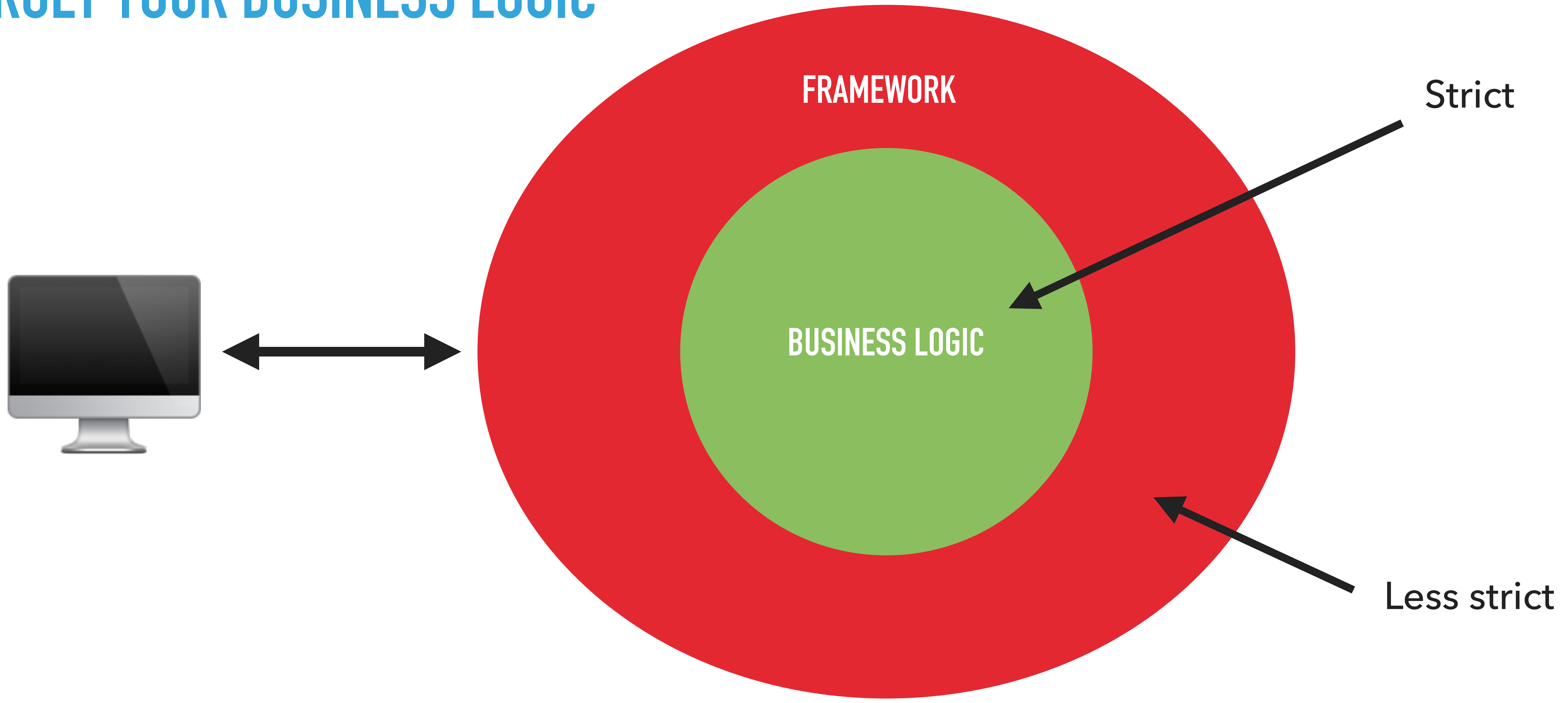
TARGET YOUR BUSINESS LOGIC



TARGET YOUR BUSINESS LOGIC



TARGET YOUR BUSINESS LOGIC



ADAPTORS FOR 3RD PARTY LIBRARIES: PROBLEM

```
interface Hasher {

   /**
    * @return string
    */
   public function encode();

}

… in our code …

$hash = $this->hasher->encode($id);
```

ADAPTORS FOR 3RD PARTY LIBRARIES: PROBLEM

```
interface Hasher {

   /**
    * @return string
    */
   public function encode();

}

… in our code …

$hash = $this->hasher->encode($id);
```

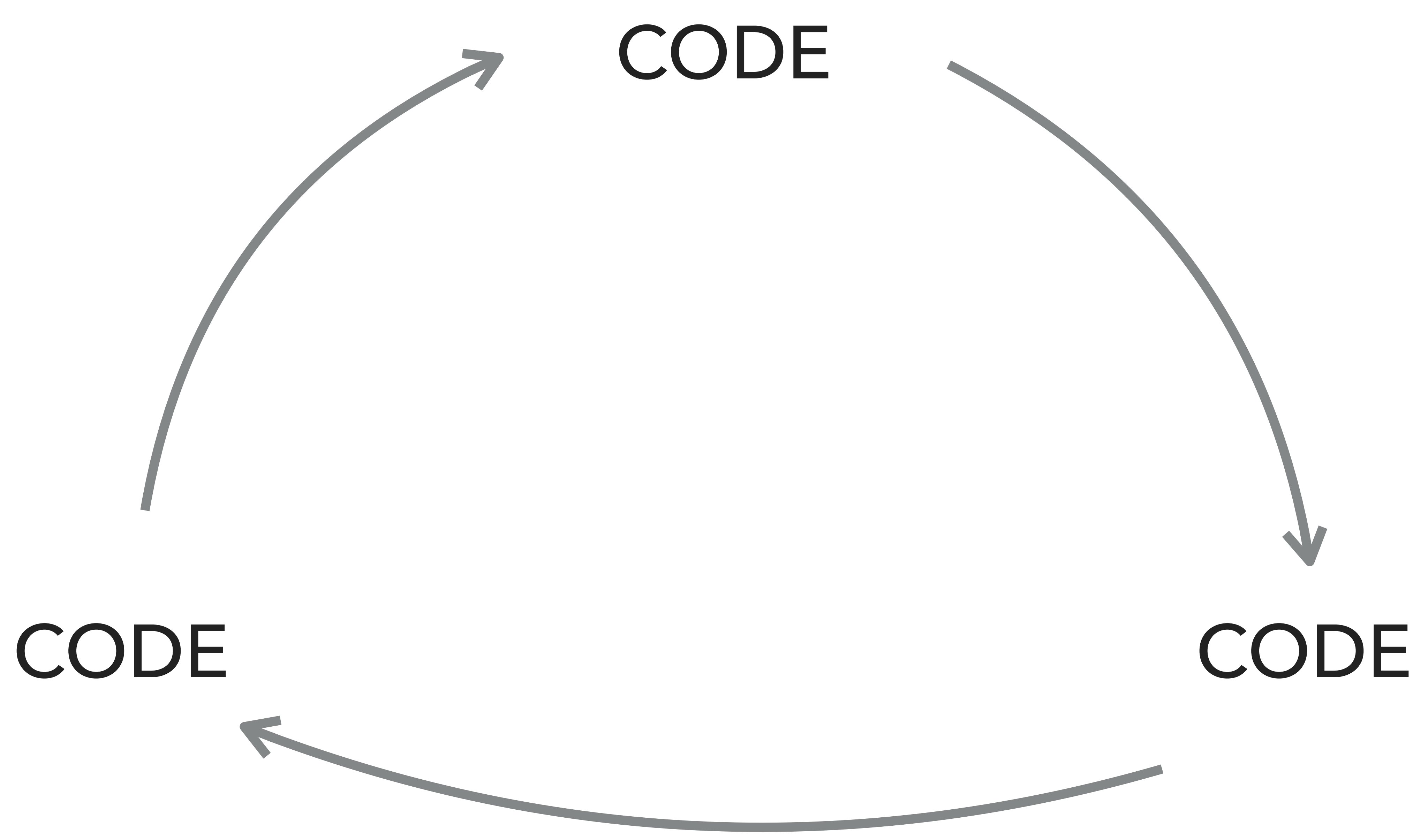
ADAPTORS FOR 3RD PARTY LIBRARIES: A SOLUTION

```
class CleanHasher {

  /** @var Hasher $hasher */
  private $hasher;

  … constructor to inject Hasher …

  public function encode(int $id): string {
      return $this->hasher->encode($id);
  }
}


… in our code …

$hash = $this->cleanHasher->encode($id);
```

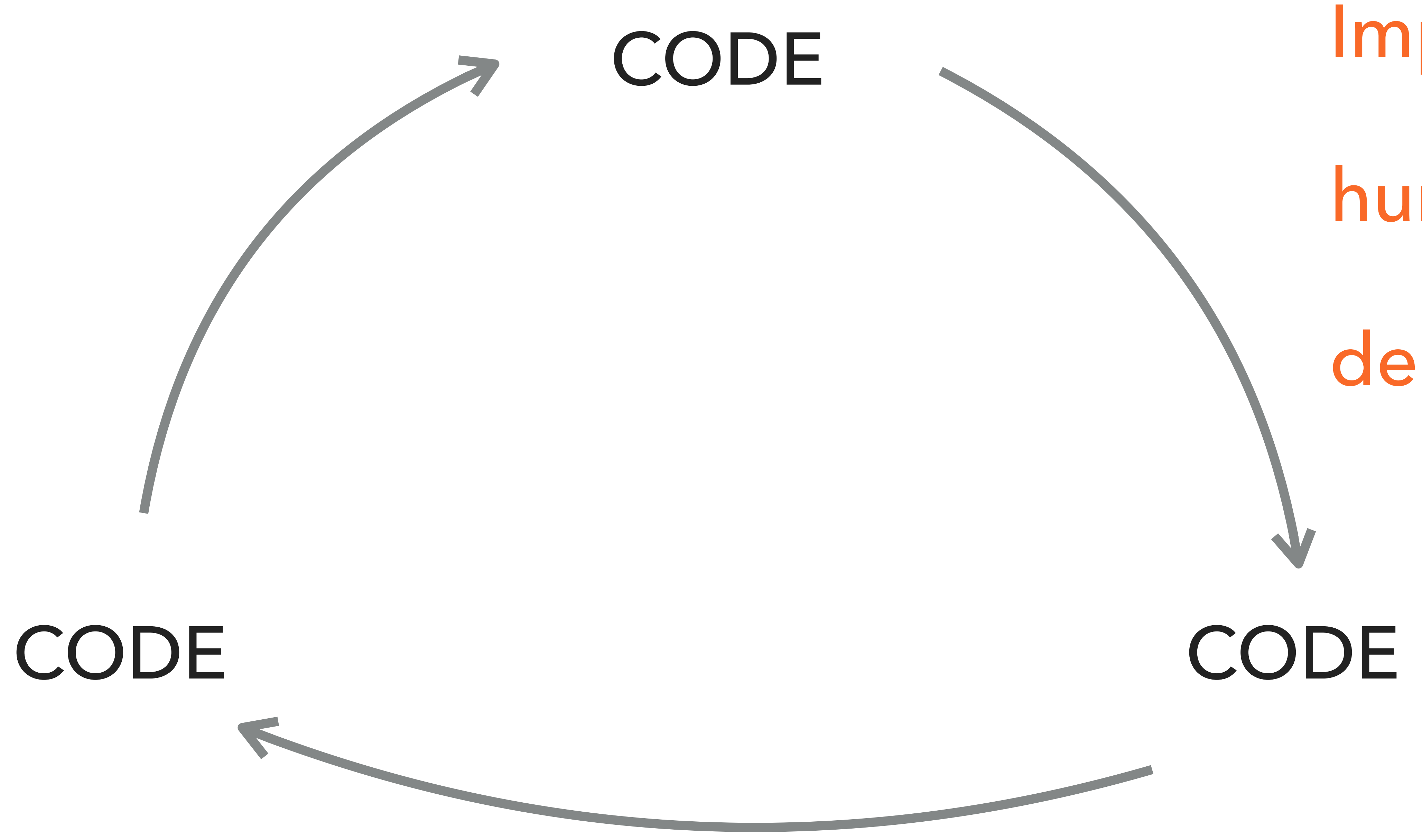
LEARN FROM MISTAKES AND DON'T BE SLOPPY

- Architect code better
- Type hint properties
- Type hint closures (including return)
- Use void if method doesn't return anything

IS THIS WORTH IT?

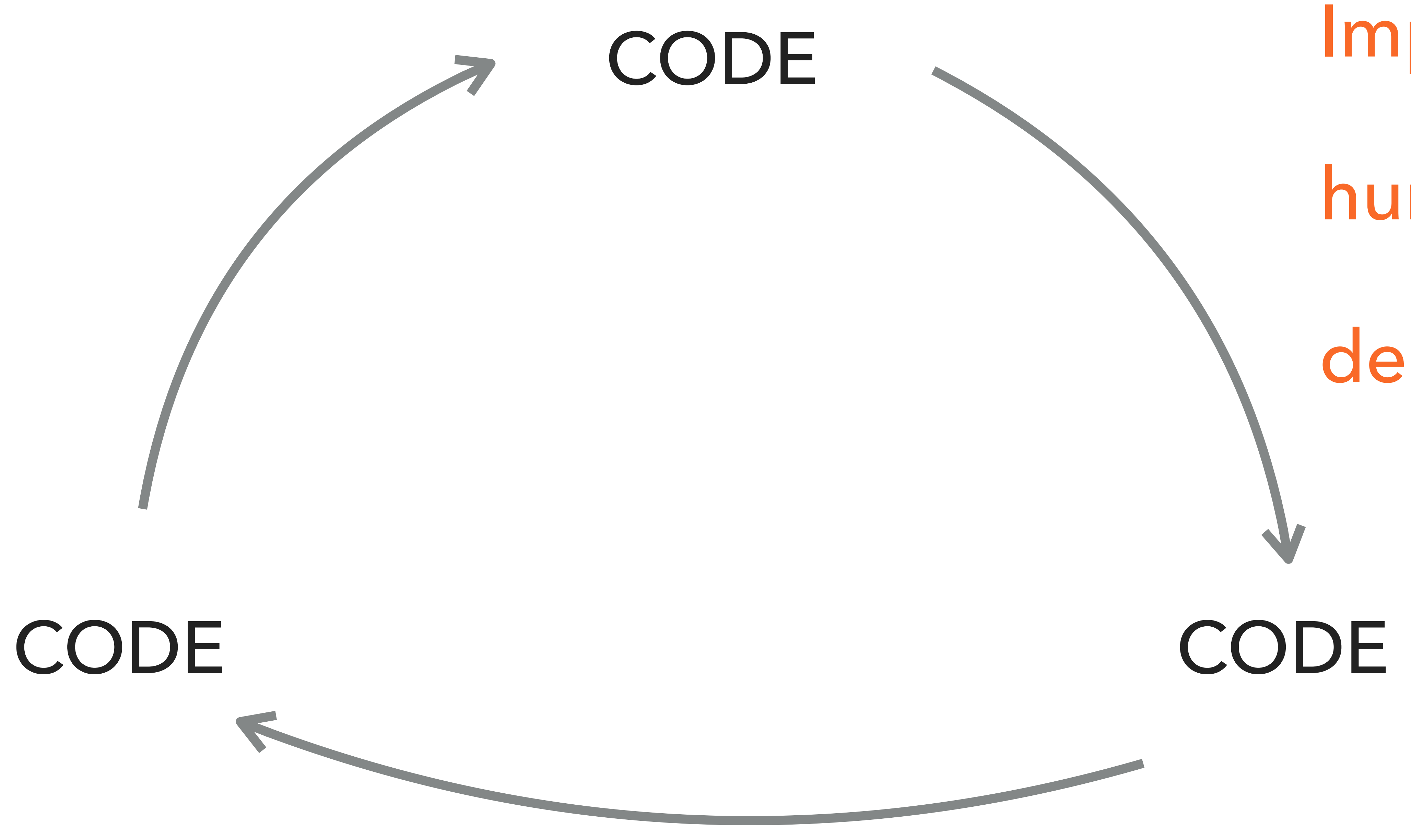


IS THIS WORTH IT?


$$\begin{tikzcd}
& \text{CODE} \arrow[dr, bend left] & \\
\text{CODE} \arrow[ur, bend left] & & \text{CODE} \arrow[ll, bend left]
\end{tikzcd}$$

Improve code so that both humans and static analysis have deeper understanding

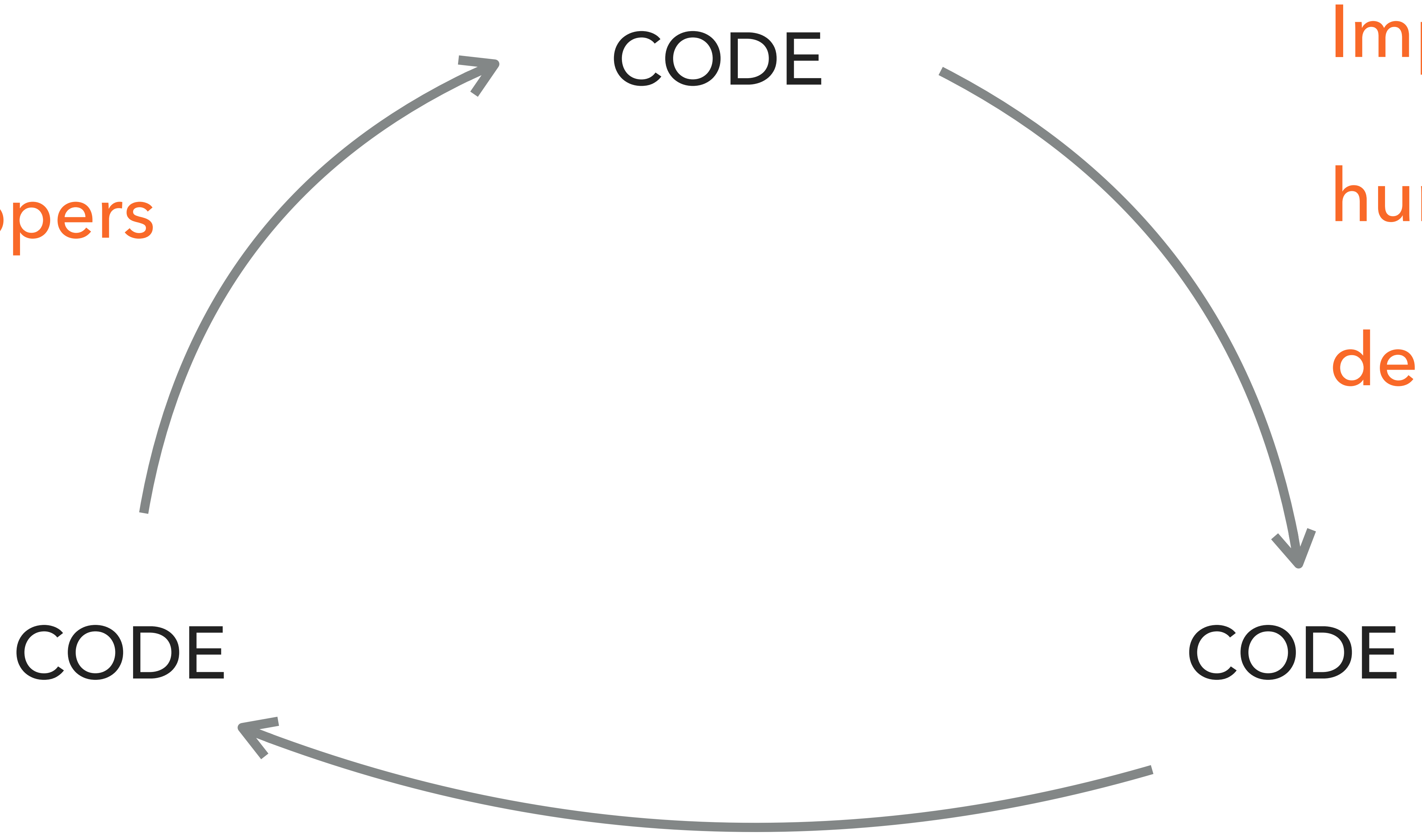
IS THIS WORTH IT?



Improve code so that both humans and static analysis have deeper understanding

Use IDE to do refactors.

IS THIS WORTH IT?



Improve code so that both humans and static analysis have deeper understanding

Use IDE to do refactors.

Code easier for developers to understand

APPROPRIATE APPLICATION OF STATIC ANALYSIS REDUCES THE OVERALL COST OF SOFTWARE DEVELOPMENT.

CHAPTER 7:

CHAPTER 7: BASELINE STATIC ANALYSIS RESULTS

CHAPTER 7: BASELINE STATIC ANALYSIS RESULTS

Problem

Problem

Problem

Problem

Problem

CHAPTER 7: BASELINE STATIC ANALYSIS RESULTS

Problem

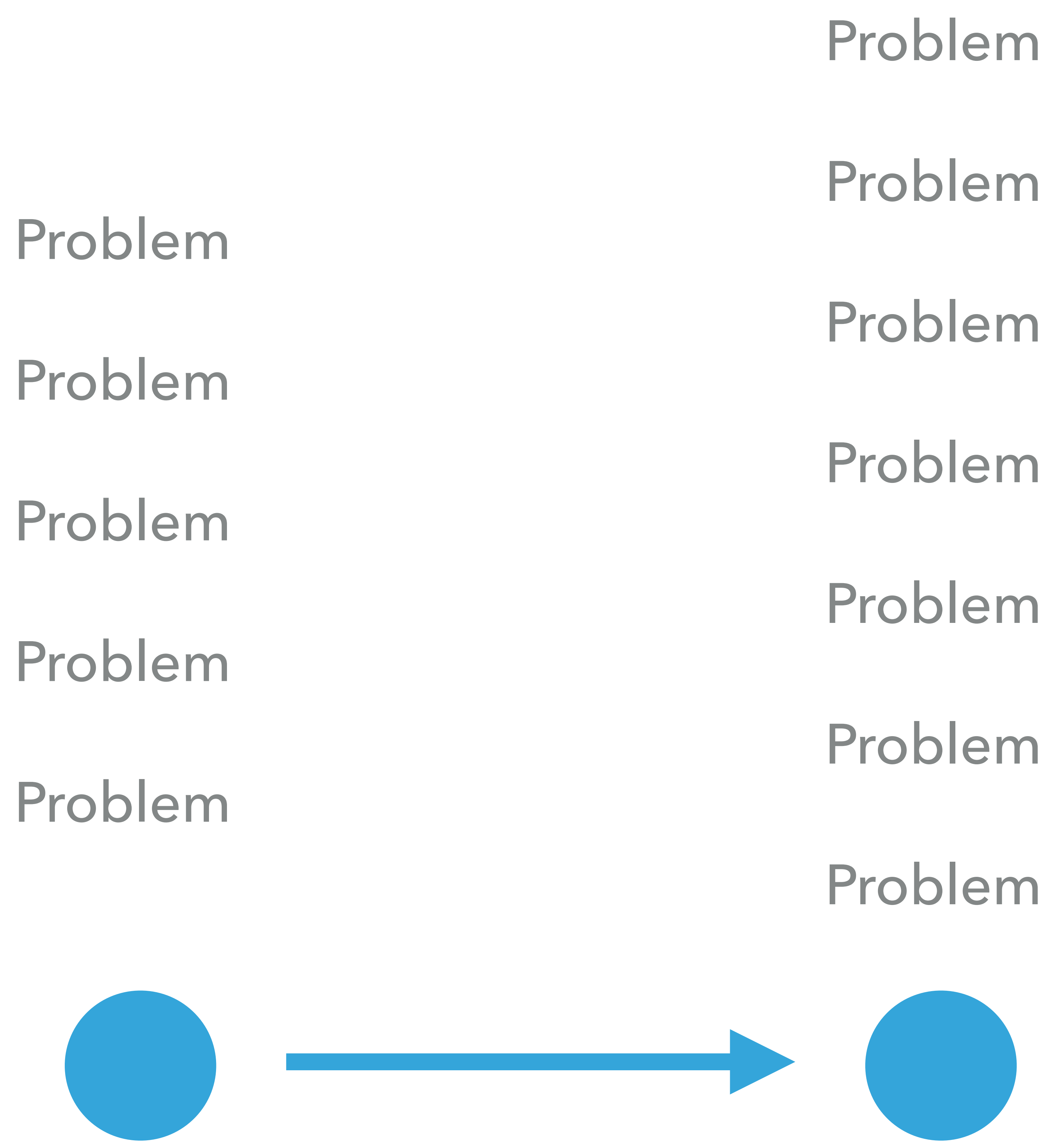
Problem

Problem

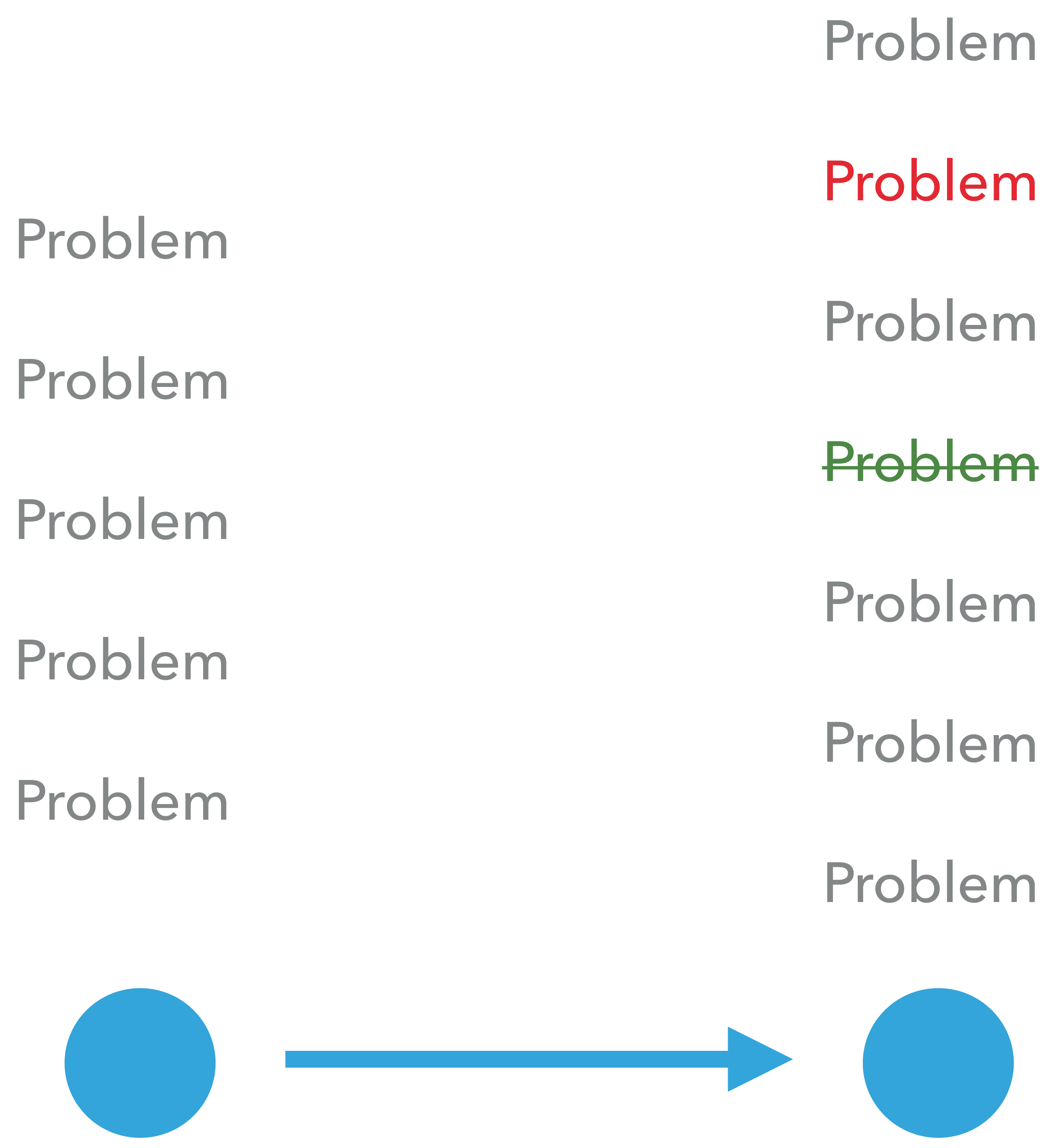
Problem

Problem

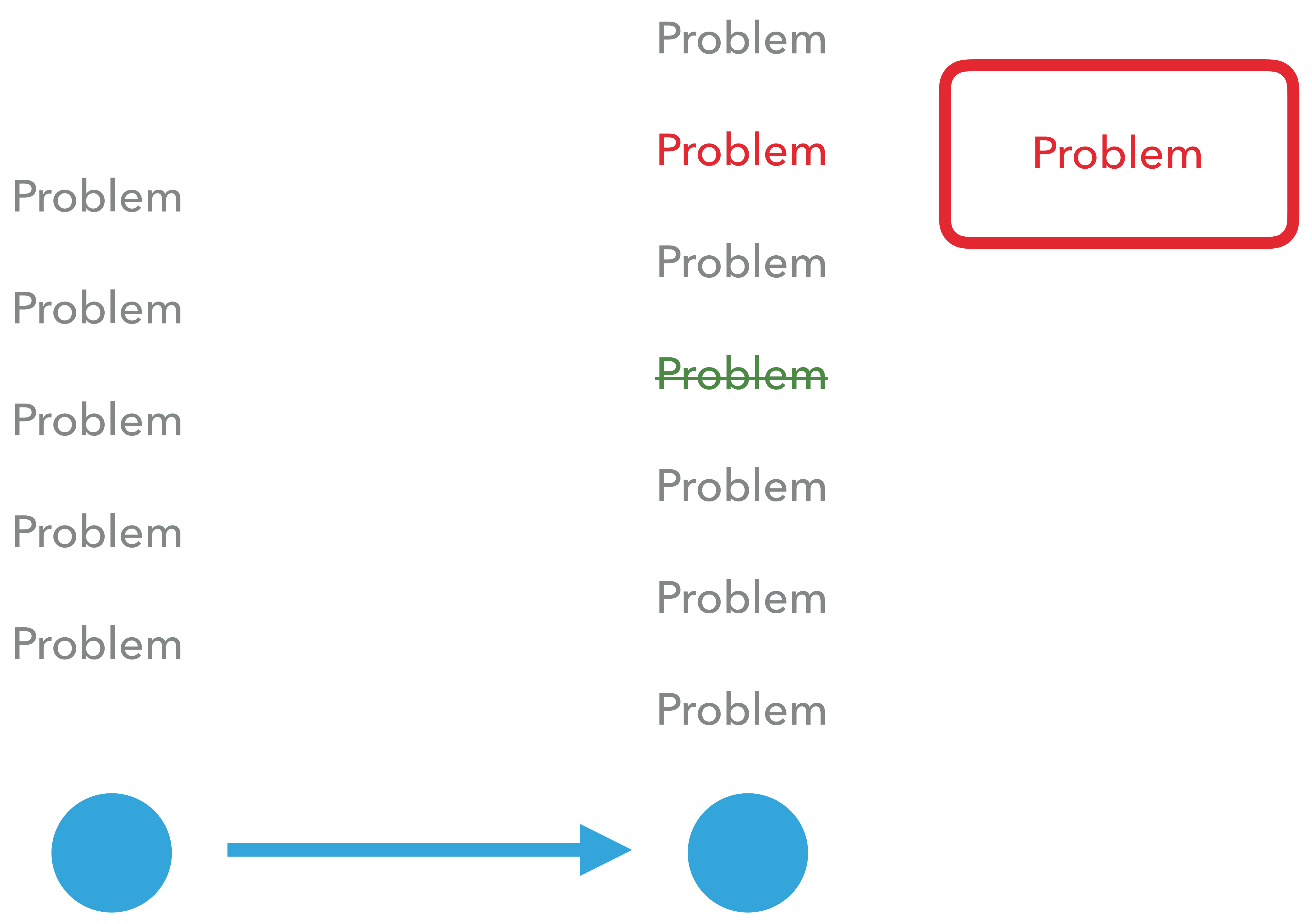
CHAPTER 7: BASELINE STATIC ANALYSIS RESULTS



CHAPTER 7: BASELINE STATIC ANALYSIS RESULTS



CHAPTER 7: BASELINE STATIC ANALYSIS RESULTS



STATIC ANALYSIS RESULTS BASELINE (SARB)

- Available soon: https://github.com/DaveLiddament/sarb
    - Supports:
        - Psalm, PHPStan, Phan
        - Easy to add more static analysis tools. Don’t need to be for PHP.
    - Requires repo uses git

SARB: CREATE BASELINE

```
# Run Psalm on the code, output is psalm_output.json

> sarb create-baseline \
   --results-format=psalm-json \
   --project-dir=~/project/acme \
   --results-file=psalm_output.json \
   --baseline-file=baseline.json

Baseline created with 328 problems.

>
```

SARB: REMOVE BASELINE FROM RESULTS

```
# Run Psalm on the updated code, output is psalm_output.json

> sarb remove-baseline-results \
   --results-format=psalm-json \
   --project-dir=~/project/acme \
   --results-file=psalm_output.json \
   --baseline-file=baseline.json \
   --output-file=filtered_results.json


Original results contained 334 problems.
Baseline contained 328 problems.
After baseline removed there are 15 new problems.
>
```

SARB: REMOVE BASELINE FROM RESULTS

```
# Run Psalm on the updated code, output is psalm_output.json

> sarb remove-baseline-results \
   --results-format=psalm-json \
   --project-dir=~/project/acme \
   --results-file=psalm_output.json \
   --baseline-file=baseline.json \
   --output-file=filtered_results.json


Original results contained 334 problems.
Baseline contained 328 problems.
After baseline removed there are 15 new problems.
>
```

SARB: REMOVE BASELINE FROM RESULTS

```
# Run Psalm on the updated code, output is psalm_output.json

> sarb remove-baseline-results \
   --results-format=psalm-json \
   --project-dir=~/project/acme \
   --results-file=psalm_output.json \
   --baseline-file=baseline.json \
   --output-file=filtered_results.json


Original results contained 334 problems.
Baseline contained 328 problems.
After baseline removed there are 15 new problems.
>
```

SARB BEHIND THE SCENES: BASELINE

Type: psalm-json **History Marker:** 06b982c6b3d15ef1eae827038d9d2bcb0ae71329

Type	File	Line number
InvalidNullableReturnType	src/Entity/Person.php	93
PossiblyNullReference	src/Entity/Shop.php	57
InvalidScalarArgument	src/Purchase/Begin.php	126

SARB BEHIND THE SCENES: REMOVING THE BASELINE RESULTS

SARB BEHIND THE SCENES: REMOVING THE BASELINE RESULTS

- **Problem:** InvalidNullableReturnType src/Entity/Employee.php:73

SARB BEHIND THE SCENES: REMOVING THE BASELINE RESULTS

- **Problem:** InvalidNullableReturnType src/Entity/Employee.php:73
- What is the location of src/Entity/Employee.php:73 at the baseline?

SARB BEHIND THE SCENES: REMOVING THE BASELINE RESULTS

- **Problem:** InvalidNullableReturnType src/Entity/Employee.php:73
- What is the location of src/Entity/Employee.php:73 at the baseline?
- History Analyser says: src/Entity/Person.php:93

SARB BEHIND THE SCENES: REMOVING THE BASELINE RESULTS

- **Problem:** InvalidNullableReturnType src/Entity/Employee.php:73
- What is the location of src/Entity/Employee.php:73 at the baseline?
- History Analyser says: src/Entity/Person.php:93
- Did we have a problem InvalidNullableReturnType at src/Entity/Person.php:93 in the baseline?

SARB BEHIND THE SCENES: BASELINE

Type: psalm-json **History Marker:** 06b982c6b3d15ef1eae827038d9d2bcb0ae71329

Type	File	Line number
InvalidNullableReturnType	src/Entity/Person.php	93
PossiblyNullReference	src/Entity/Shop.php	57
InvalidScalarArgument	src/Purchase/Begin.php	126

SARB BEHIND THE SCENES: BASELINE

Type: psalm-json **History Marker:** 06b982c6b3d15ef1eae827038d9d2bcb0ae71329

Type	File	Line number
InvalidNullableReturnType	src/Entity/Person.php	93
PossiblyNullReference	src/Entity/Shop.php	57
InvalidScalarArgument	src/Purchase/Begin.php	126

SARB BEHIND THE SCENES: REMOVING THE BASELINE RESULTS

- **Problem:** InvalidNullableReturnType src/Entity/Employee.php:73
- What is the location of src/Entity/Employee.php:73 at the baseline?
- History Analyser says: src/Entity/Person.php:93
- Did we have a problem InvalidNullableReturnType at src/Entity/Person.php:93 in the baseline?
- Yes so don't report this problem.

STATIC ANALYSIS WITH SARB

STATIC ANALYSIS WITH SARB

- Run static analysis tool

STATIC ANALYSIS WITH SARB

- Run static analysis tool
- Fix all bugs you decide need fixing

STATIC ANALYSIS WITH SARB

- Run static analysis tool
- Fix all bugs you decide need fixing
- Run static analysis tool again

STATIC ANALYSIS WITH SARB

- Run static analysis tool
- Fix all bugs you decide need fixing
- Run static analysis tool again
- Generate baseline

STATIC ANALYSIS WITH SARB

- Run static analysis tool
- Fix all bugs you decide need fixing
- Run static analysis tool again
- Generate baseline
- Repeat forever:
    - Write code
    - Run analysis
    - Remove baseline results
    - Fix bugs

STATIC ANALYSIS WITH SARB

- Run static analysis tool
- Fix all bugs you decide need fixing
- Run static analysis tool again
- Generate baseline
- Repeat forever:
  - Write code
  - Run analysis
  - Remove baseline results
  - Fix bugs

WHAT AN ADVENTURE IT HAS BEEN…

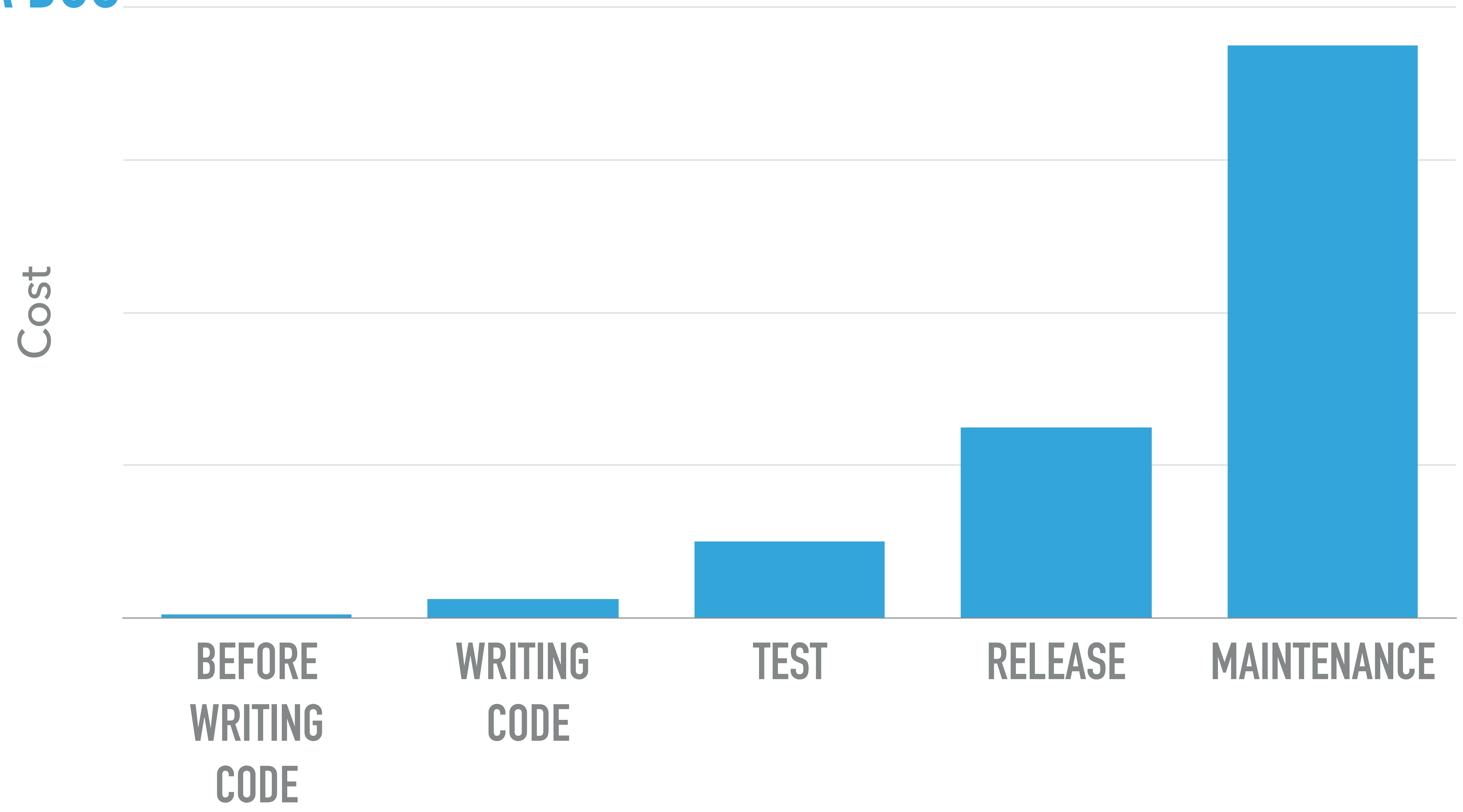
WHAT AN ADVENTURE IT HAS BEEN…

APPROPRIATE APPLICATION OF STATIC ANALYSIS REDUCES THE OVERALL COST OF SOFTWARE DEVELOPMENT.

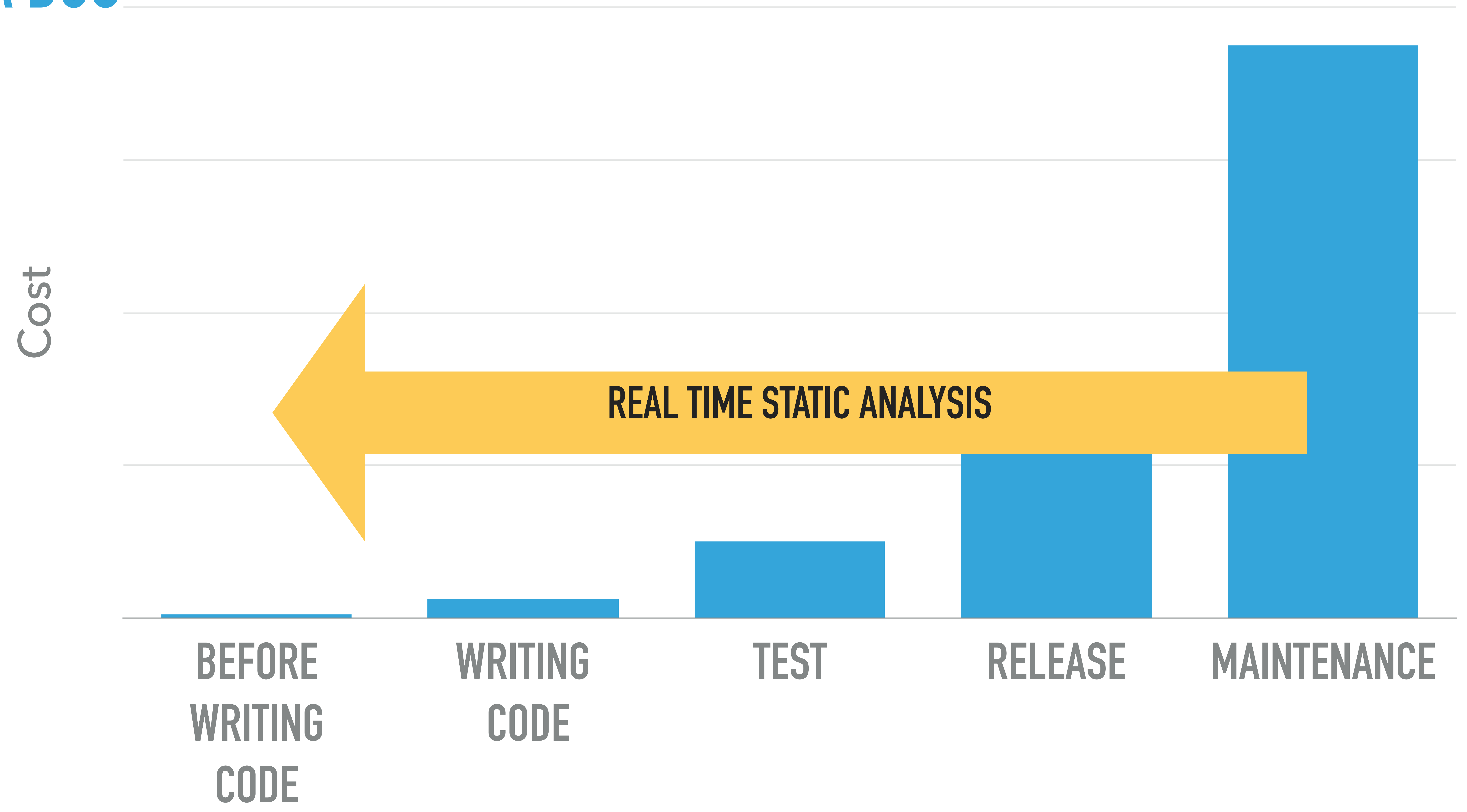
Static analysis tells you that your code is incorrect.

Tests tell you a particular scenario is working correctly.

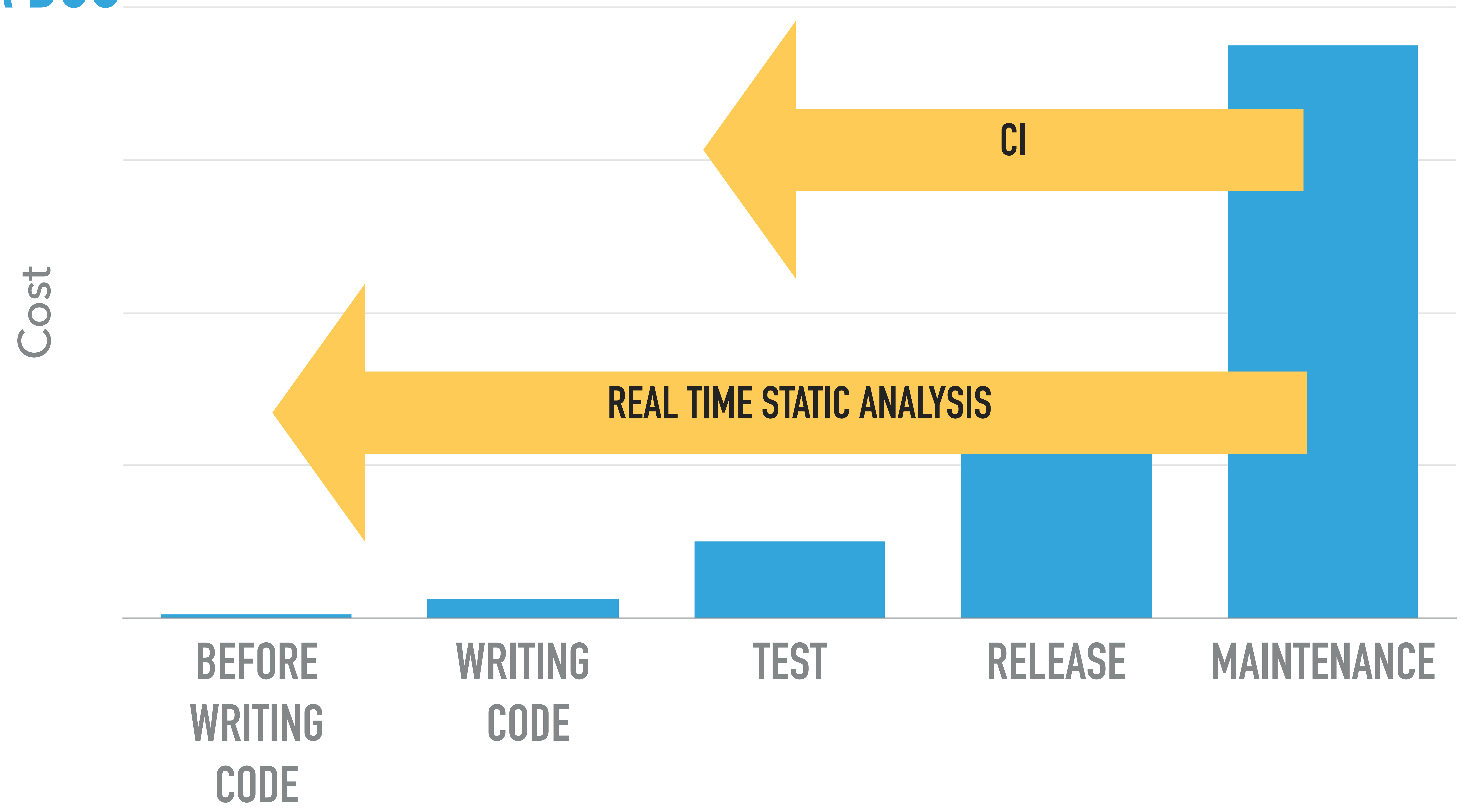
COST OF A BUG


Cost
BEFORE WRITING CODE
WRITING CODE
TEST
RELEASE
MAINTENANCE

COST OF A BUG


Cost
REAL TIME STATIC ANALYSIS
BEFORE WRITING CODE
WRITING CODE
TEST
RELEASE
MAINTENANCE

COST OF A BUG


Cost
CI
REAL TIME STATIC ANALYSIS
BEFORE WRITING CODE
WRITING CODE
TEST
RELEASE
MAINTENANCE

CI TOOLSET

- Composer validate: `composer validate --strict`
- Parallel lint: `jakub-onderka/php-parallel-lint`
- PHP CS fixer: `friendsofsymfony/php-cs-fixer`
- Var dump checker: `jakub-onderka/php-var-dump-checker`
- Security checker: `sensiolabs/security-checker`

PHP bible for static analysis tools: https://github.com/exakat/php-static-analysis-tools

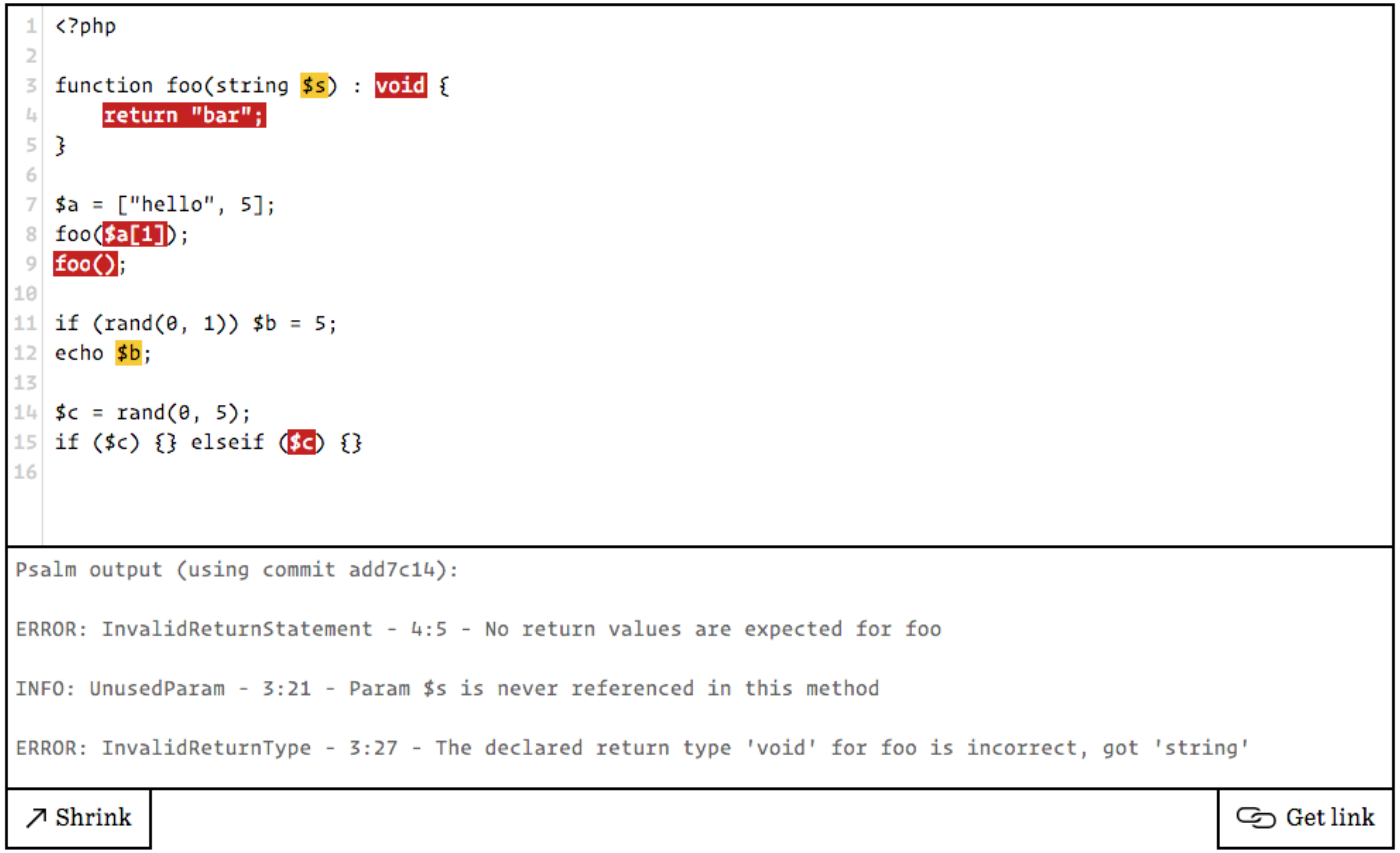
REQUIREMENTS FOR REAL TIME STATIC ANALYSIS TOOL (IDE)

- Understand entire codebase (including vendor directory)
- Highlight errors in real time
- Suggest / autocomplete based on context
- Refactoring (e.g. rename, move, extract)

REQUIREMENTS FOR REAL TIME STATIC ANALYSIS TOOL (IDE)

- Understand entire codebase (including vendor directory)
- Highlight errors in real time
- Suggest / autocomplete based on context
- Refactoring (e.g. rename, move, extract)

USE ADVANCED STATIC ANALYSIS TOOLS IN CI



```php
<?php

function foo(string $s) : void {
    return "bar";
}

$a = ["hello", 5];
foo($a[1]);
foo();

if (rand(0, 1)) $b = 5;
echo $b;

$c = rand(0, 5);
if ($c) {} elseif ($c) {}

```

```
Psalm output (using commit add7c14):

ERROR: InvalidReturnStatement - 4:5 - No return values are expected for foo

INFO: UnusedParam - 3:21 - Param $s is never referenced in this method

ERROR: InvalidReturnType - 3:27 - The declared return type 'void' for foo is incorrect, got 'string'
```

SUMMARY

@daveliddament

THANK YOU FOR LISTENING

SUMMARY

FEEDBACK

- Please, please, please give feedback….
- You can win a PHPStorm licence
- DMs are open on twitter @daveliddament
- I'd like to know…
  - 1 thing you liked (optional)
  - Advice on at least 1 way it could be improved

REFERENCES

- [1] Mika V. Mantyla and Casper Lassenius "What Types of Defects Are Really Discovered in Code Reviews?" IEEE Transactions on Software Engineering
- [2] Harvey Siy, Lawrence Votta "Does The Modern Code Inspection Have Value?"
- [3] R.K. Bandi, V.K. Vaishnavi, and D.E. Turk, "Predicting Maintenance Performance Using Object-Orientated Design Complexity Metrics"

LINKS

- Static Analysis tools: https://github.com/exakat/php-static-analysis-tools
- Sample CircleCI project: https://github.com/DaveLiddament/skeleton-ci-project
- Psalm https://getpsalm.org/
- Phan: https://github.com/phan/phan
- PHPStan https://github.com/phan/phan
- Parallel lint https://github.com/JakubOnderka/PHP-Parallel-Lint
- PHP CS fixer https://github.com/FriendsOfPHP/PHP-CS-Fixer
- Var dump checker https://github.com/JakubOnderka/PHP-Var-Dump-Check
- Security checker https://security.sensiolabs.org/