



# Squash bugs with static analysis

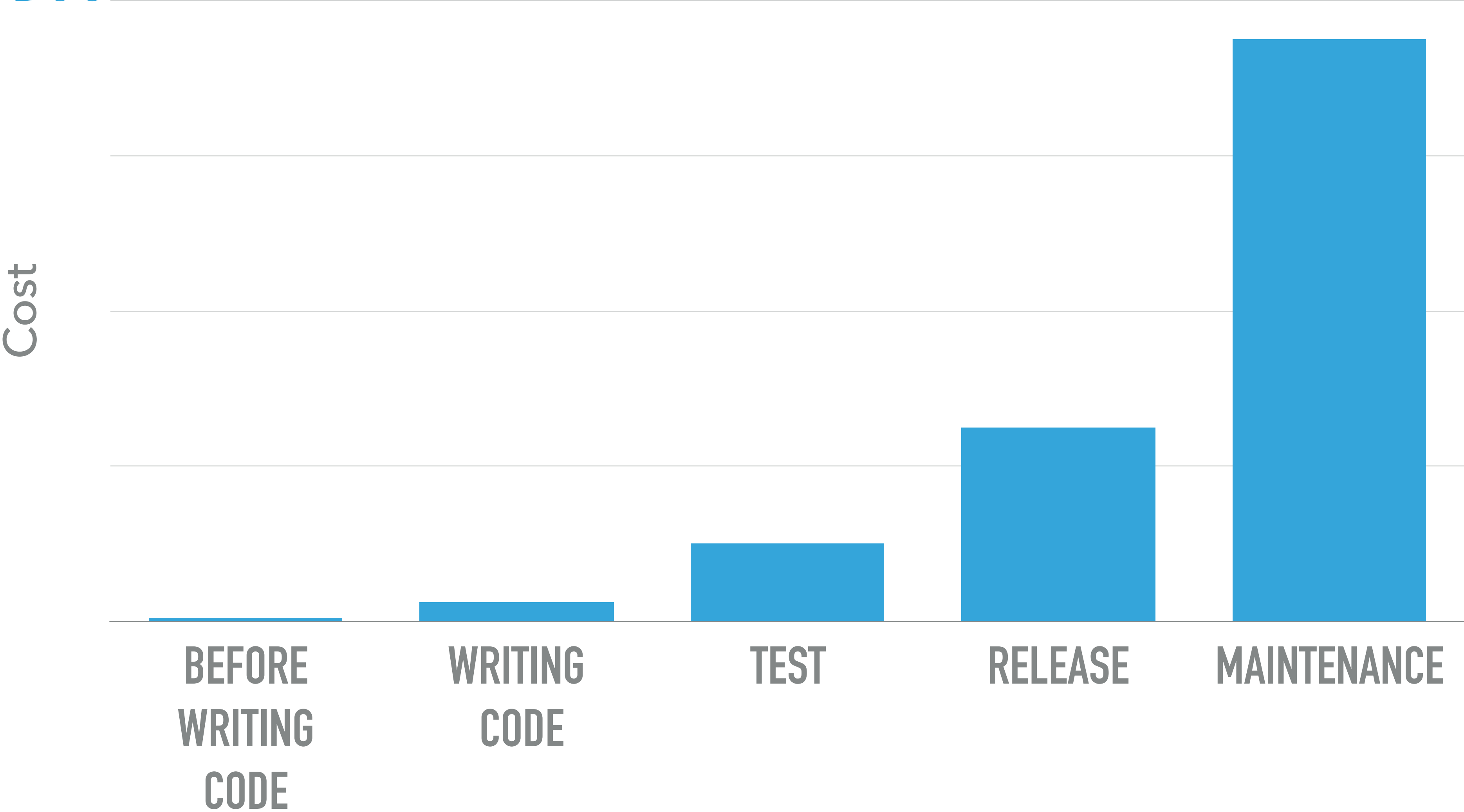
Dave Liddament

@daveliddament

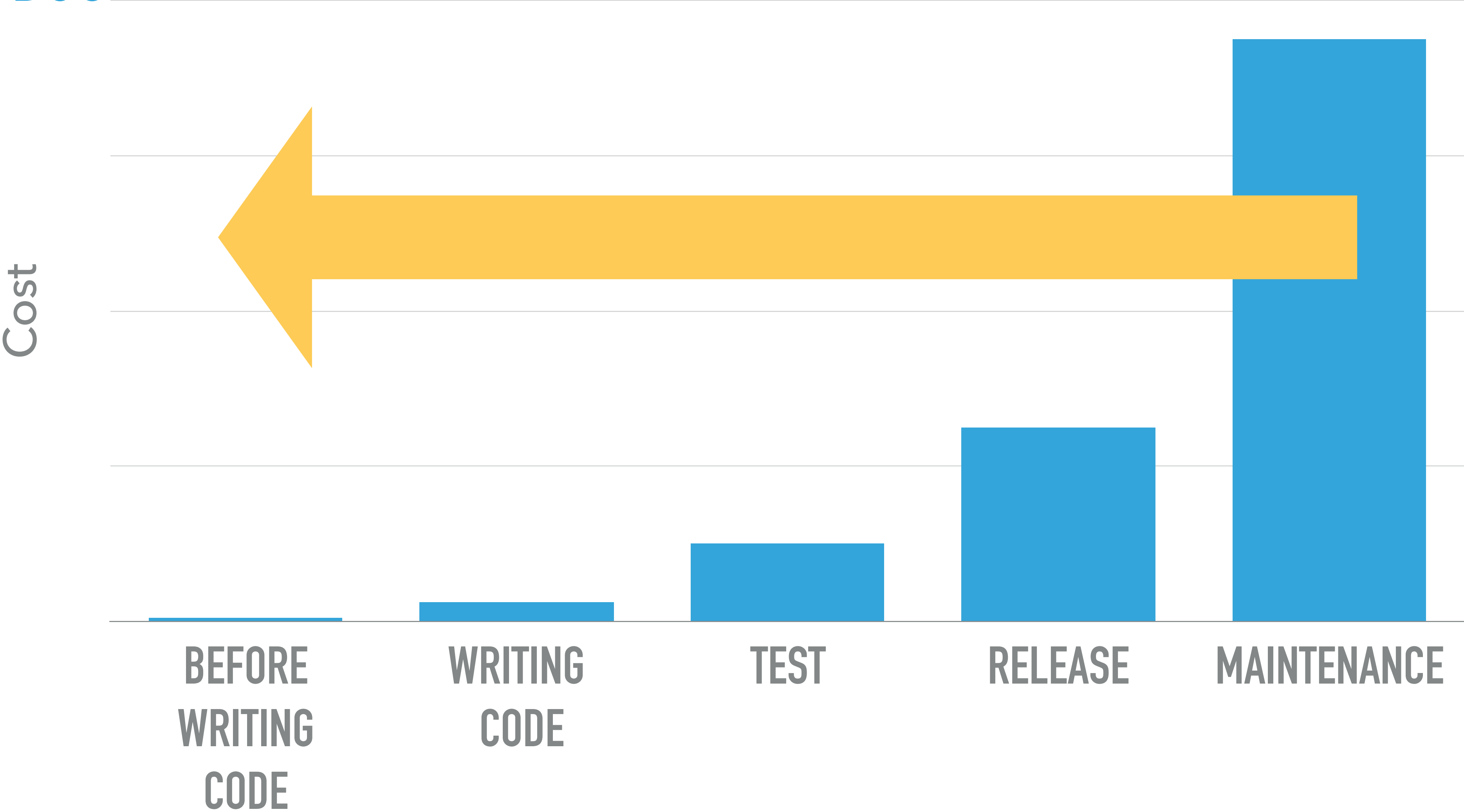


**APPROPRIATE APPLICATION OF STATIC ANALYSIS  
REDUCES THE OVERALL COST OF SOFTWARE  
DEVELOPMENT.**

# COST OF A BUG



# COST OF A BUG



### Yes

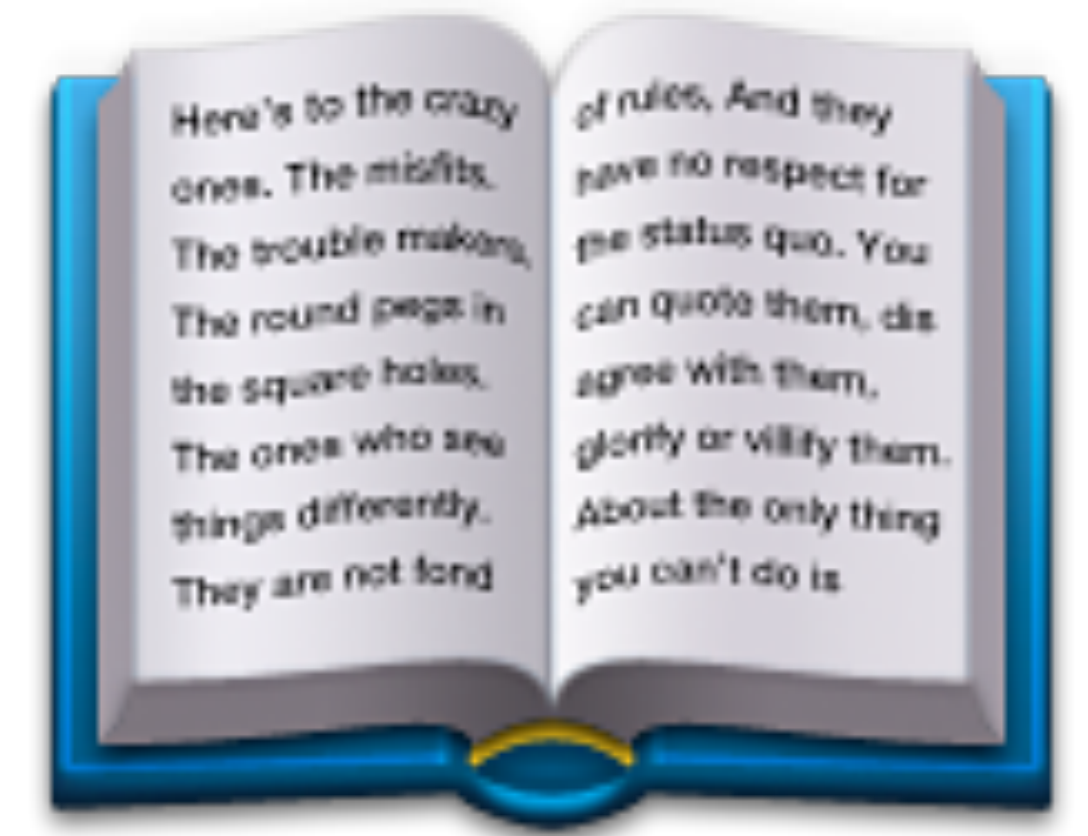
- ▶ You use no or only basic static analysis tools.
- ▶ You want to learn about more advanced tools.

### No

- ▶ You already use a powerful IDE.
- ▶ You already use tools like Phan, Psalm or PHPStan in CI.

## 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)



# 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

100 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.**

STATIC ANALYSIS

---

**MY STORY...**



## MY STORY... CHAPTER 1: CODE LOOKED LIKE THIS...

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

  <p>
    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

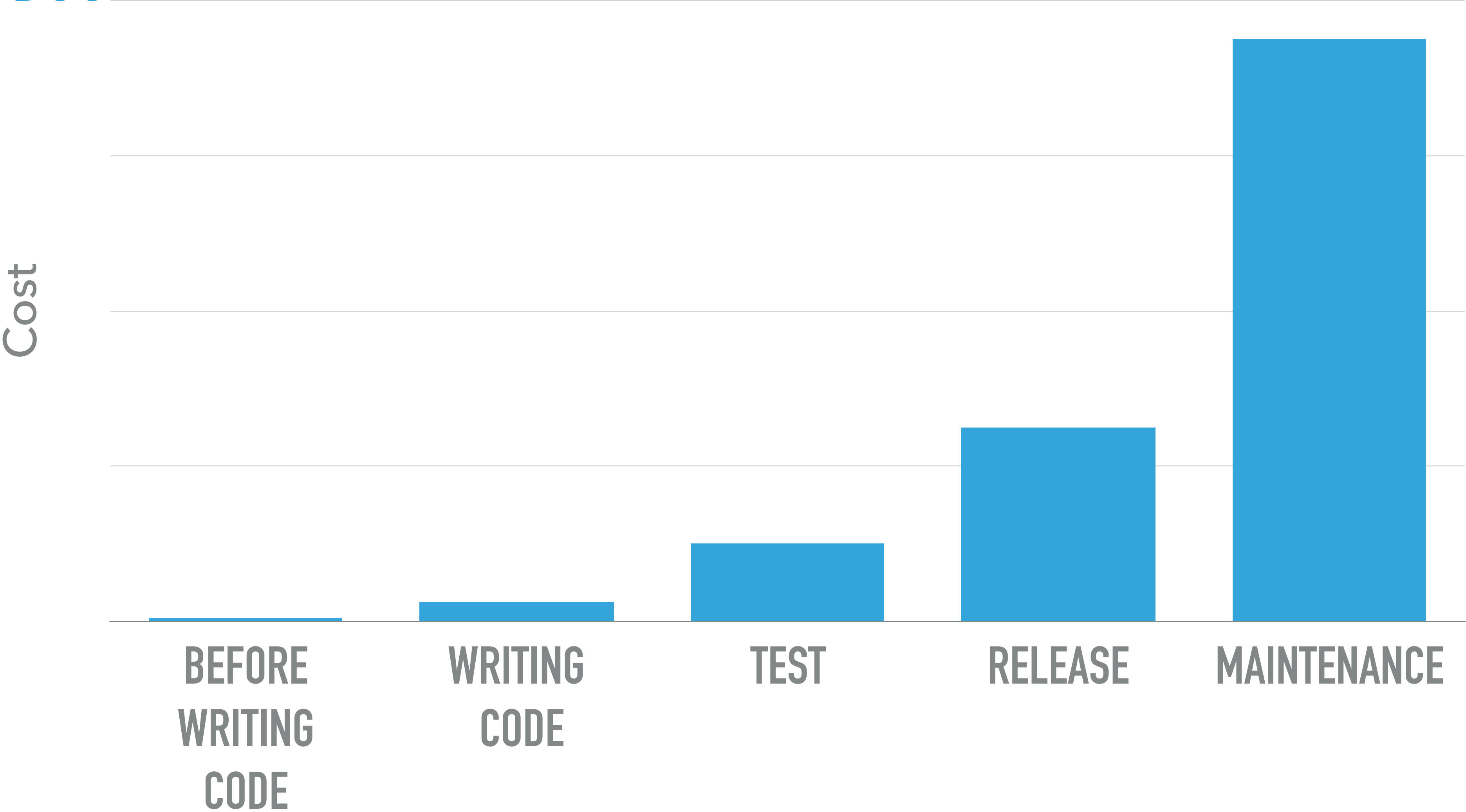
✗ No invalid syntax highlighting in editor

✗ No automated linting of code

Real time static analysis

CI

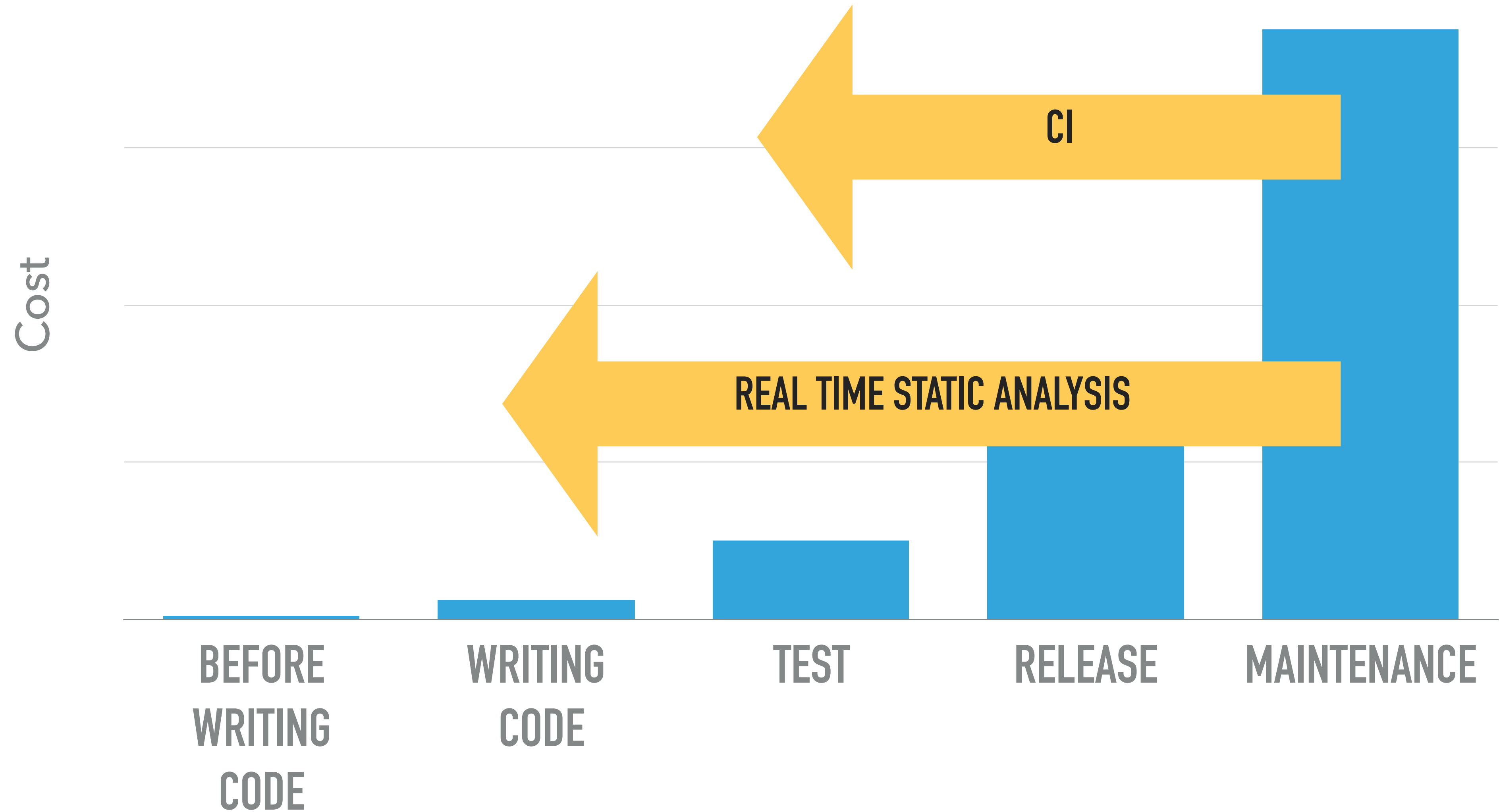
# COST OF A BUG



# COST OF A BUG



## COST OF A BUG



## 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**

## TAKE AWAY: PERFORM AUTOMATED LINTING AS PART OF CI

- ▶ Install:

- ▶ `composer require --dev jakub-onderka/php-parallel-lint`

## TAKE AWAY: PERFORM AUTOMATED LINTING AS PART OF CI

- ▶ Install:

- ▶ `composer require --dev jakub-onderka/php-parallel-lint`

- ▶ Run:

- ▶ `vendor/bin/parallel-lint src`



STATIC ANALYSIS

---

CI TOOLSET

### CI TOOLSET

- ▶ Composer validate: `composer validate --strict`

### CI TOOLSET

- ▶ Composer validate: `composer validate --strict`
- ▶ Parallel lint: `jakub-ondrka/php-parallel-lint`

### CI TOOLSET

- ▶ Composer validate: `composer validate --strict`
- ▶ Parallel lint: `jakub-ondarka/php-parallel-lint`
- ▶ PHP CS fixer: `friendsofsymfony/php-cs-fixer`

### 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-check`

### CI TOOLSET

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



### CI TOOLSET

- ▶ Composer validate: `composer validate --strict`
- ▶ Parallel lint: `jakub-ondarka/php-parallel-lint`
- ▶ PHP CS fixer: `friendsofsymfony/php-cs-fixer`
- ▶ Var dump checker: `jakub-ondarka/php-var-dump-check`
- ▶ 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`

**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 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>

### 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.



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

## AN EVOLVABILITY DEFECT

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



## AN EVOLVABILITY DEFECT

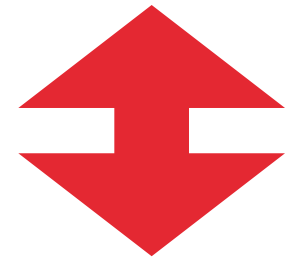
```
/**
 * @param $person
 * @return int
 */
function getAgeNextBirthday($a)
{
    return "Age next birthday " . $a->asI() + 1;
}
```

# 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: RETURN TO SOFTWARE DEVELOPER



## CHAPTER 3: TYPE HINT EVERYTHING!

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


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

```
$a = 1;
```

```
process($a);
```


Expected User, got int [more...](#) (%F1)

# GETTING THE MOST FROM REAL TIME STATIC ANALYSIS

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

user : \User

# GETTING THE MOST FROM REAL TIME STATIC ANALYSIS

 \$a = 1;

process();

user : \User



# GETTING THE MOST FROM REAL TIME STATIC ANALYSIS

```

$analysisResult->
}

return $analysisResults;

```

- getFileName()** DaveLiddament\StaticA
- toArray()** array
- getFullDetails()** string
- getLineNumber** DaveLiddament\Sta...
- isMatch**(location : \DaveLi.. bool
- getType()** string

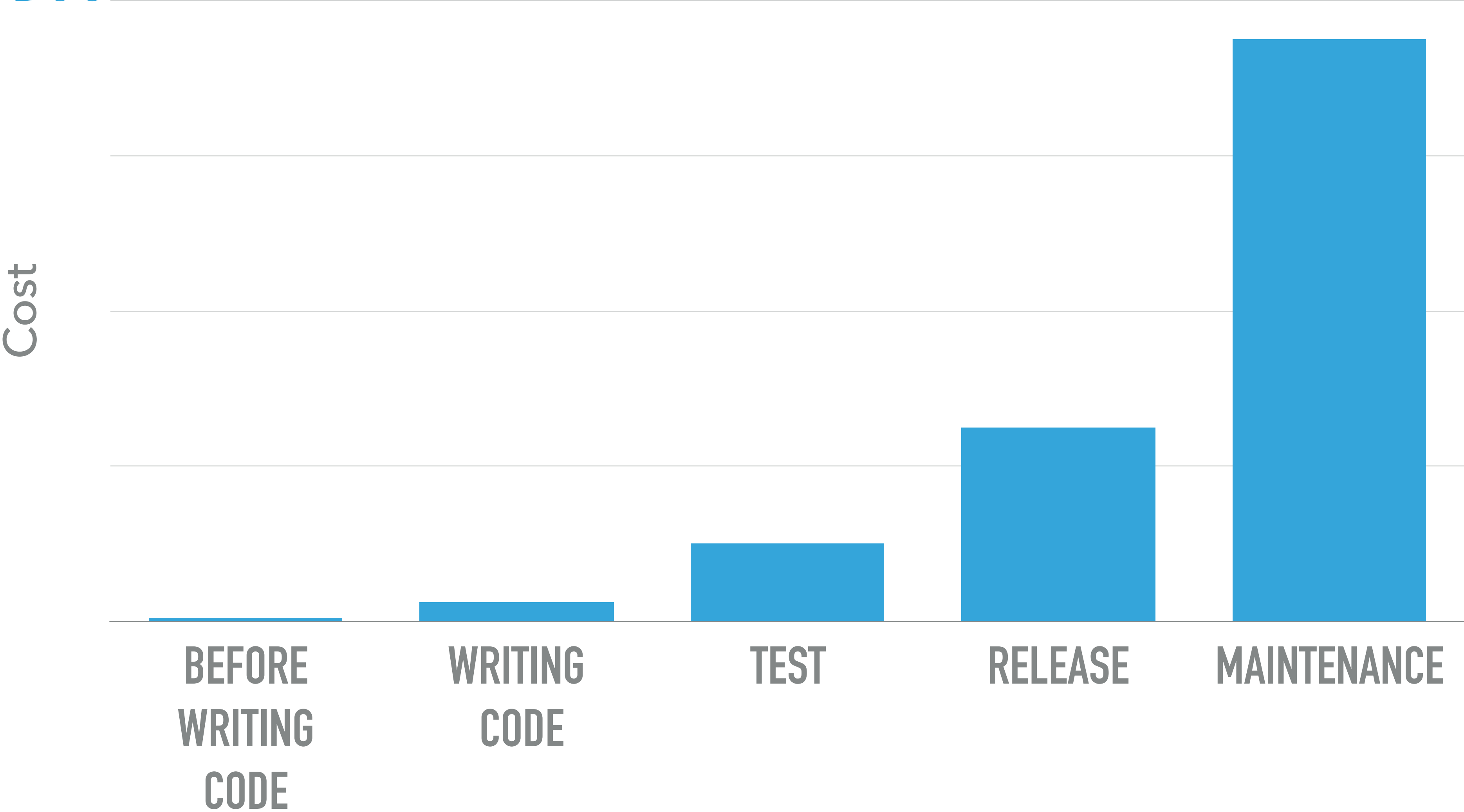
Press ^Space again to see more variants >>π



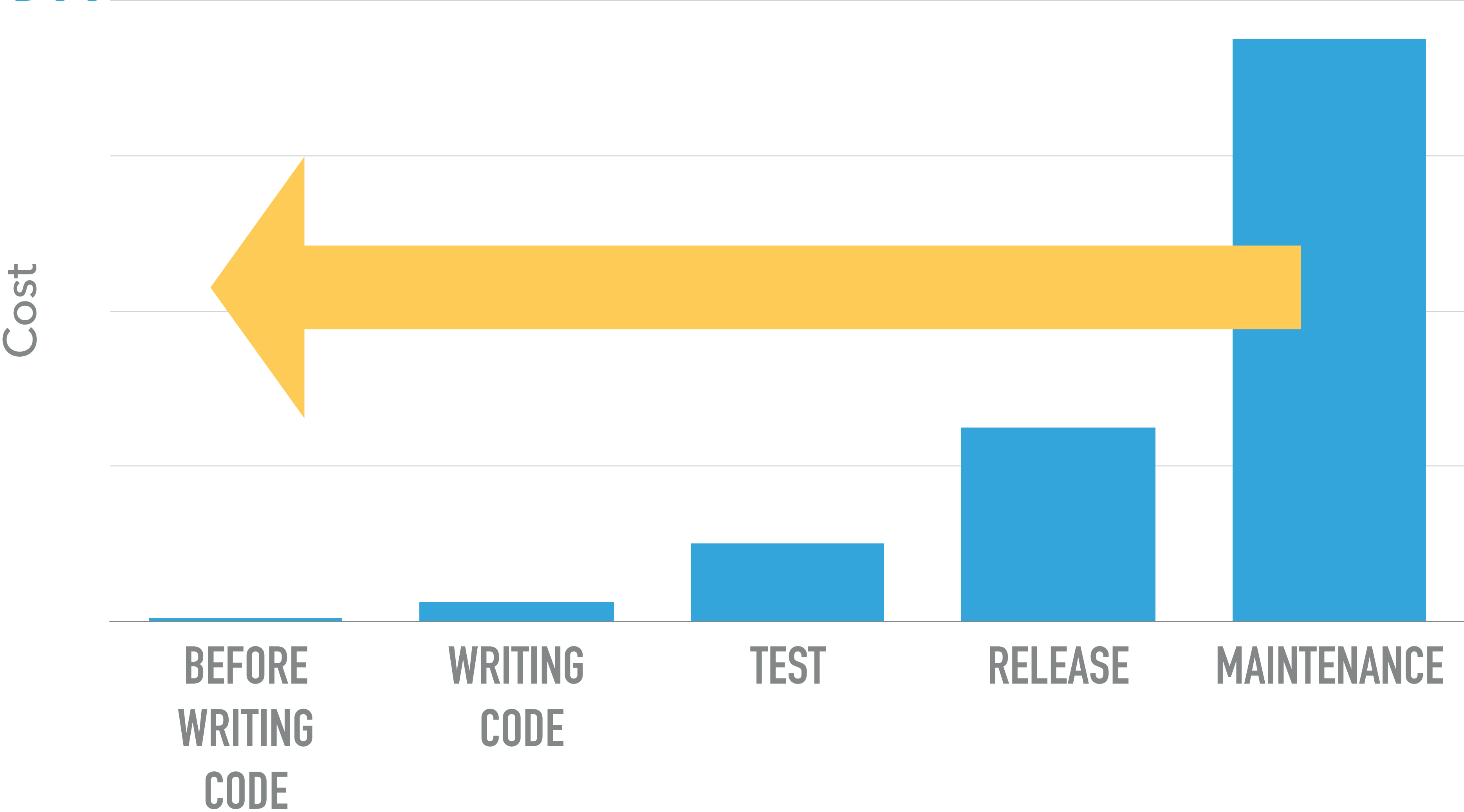
# GETTING THE MOST FROM REAL TIME STATIC ANALYSIS



COST OF A BUG



# COST OF A BUG



# REQUIREMENTS FOR REAL TIME STATIC ANALYSIS TOOL (IDE)

- ▶ Understand entire codebase
- ▶ 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 4: HAPPY



<https://github.com/DaveLiddament/skeleton-ci-project>

## STILL THIS NAGGING PROBLEM

✓ Real time static analysis

✗ CI

## CHAPTER 5: ADVANCED STATIC ANALYSIS TOOLS

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



# ADVANCED STATIC ANALYSIS TOOLS

```
1 <?php
2
3 function foo(string $s) : void {
4     return "bar";
5 }
6
7 $a = ["hello", 5];
8 foo($a[1]);
9 foo();
10
11 if (rand(0, 1)) $b = 5;
12 echo $b;
13
14 $c = rand(0, 5);
15 if ($c) {} elseif ($c) {}
16
```

Psalalm 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'

↗ Shrink

🔗 Get link

<https://getpsalm.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);  
}
```

## 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);  
  
}
```

## 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

```
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

```
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) ;  
}
```

## 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

```
18  
19 foreach($business->getEmployees() as $name => $employee) {  
20     promote($employee);  
21     welcome($name);  
22 }
```

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 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:

## 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>`

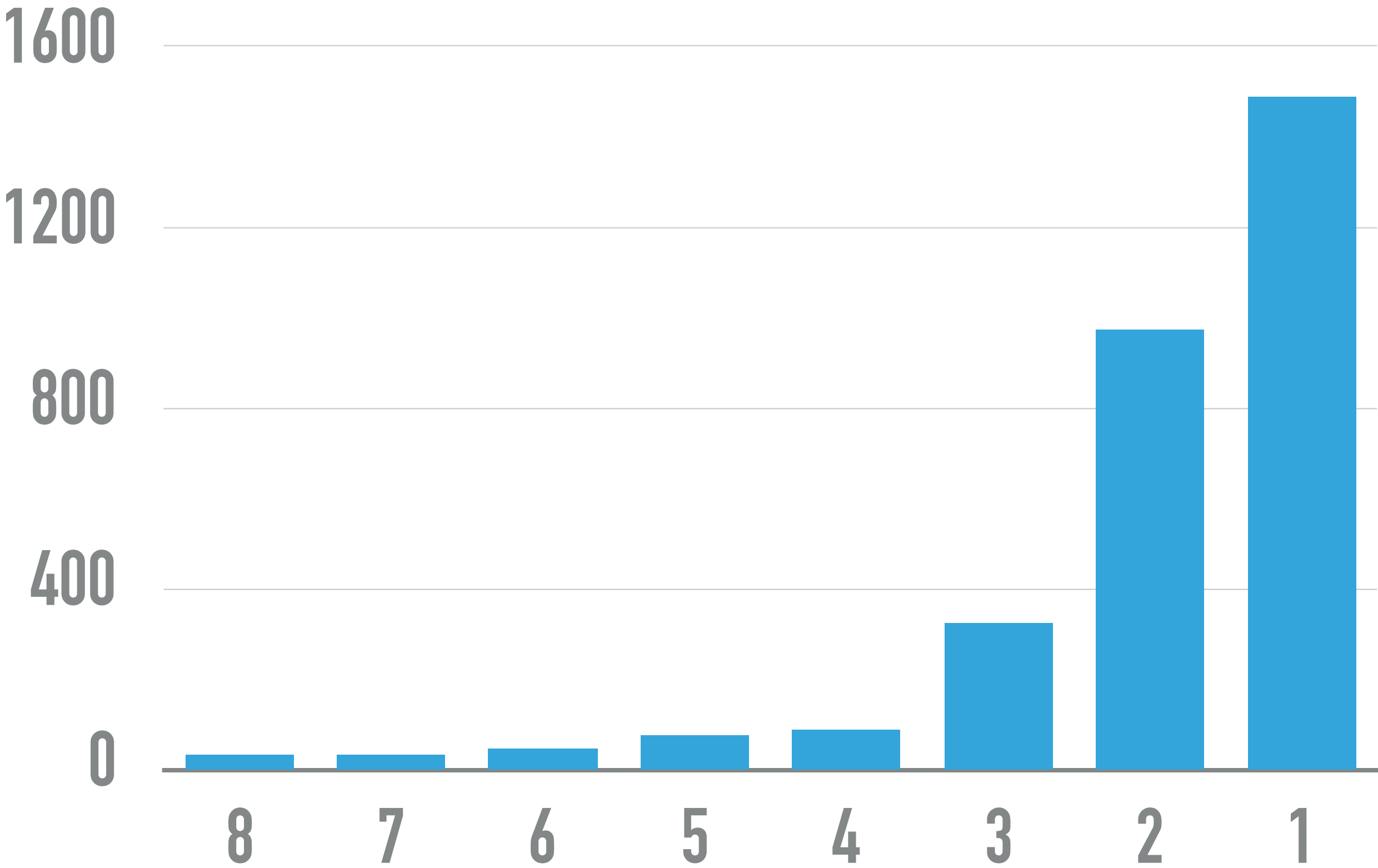
- ▶ Run:

- ▶ `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());
```

## EVOLVABILITY DEFECT

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



## EVOLVABILITY DEFECT

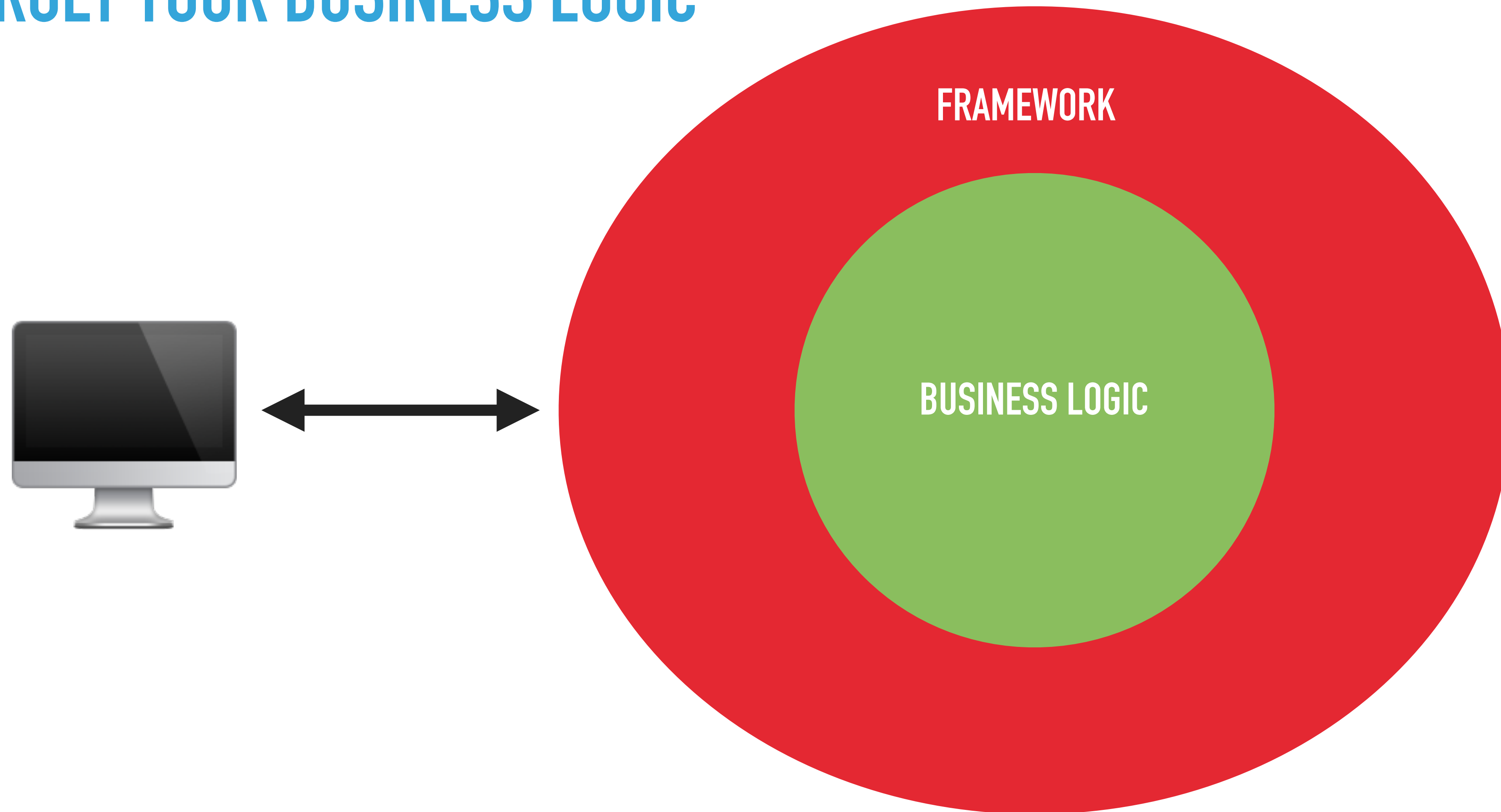
```
$plots = array_map(function(Bookmark $bookmark):Plot {  
    return $bookmark->getPlot();  
}, $bookmarks);
```

You don't really expect me to fix  
all those "bugs"?

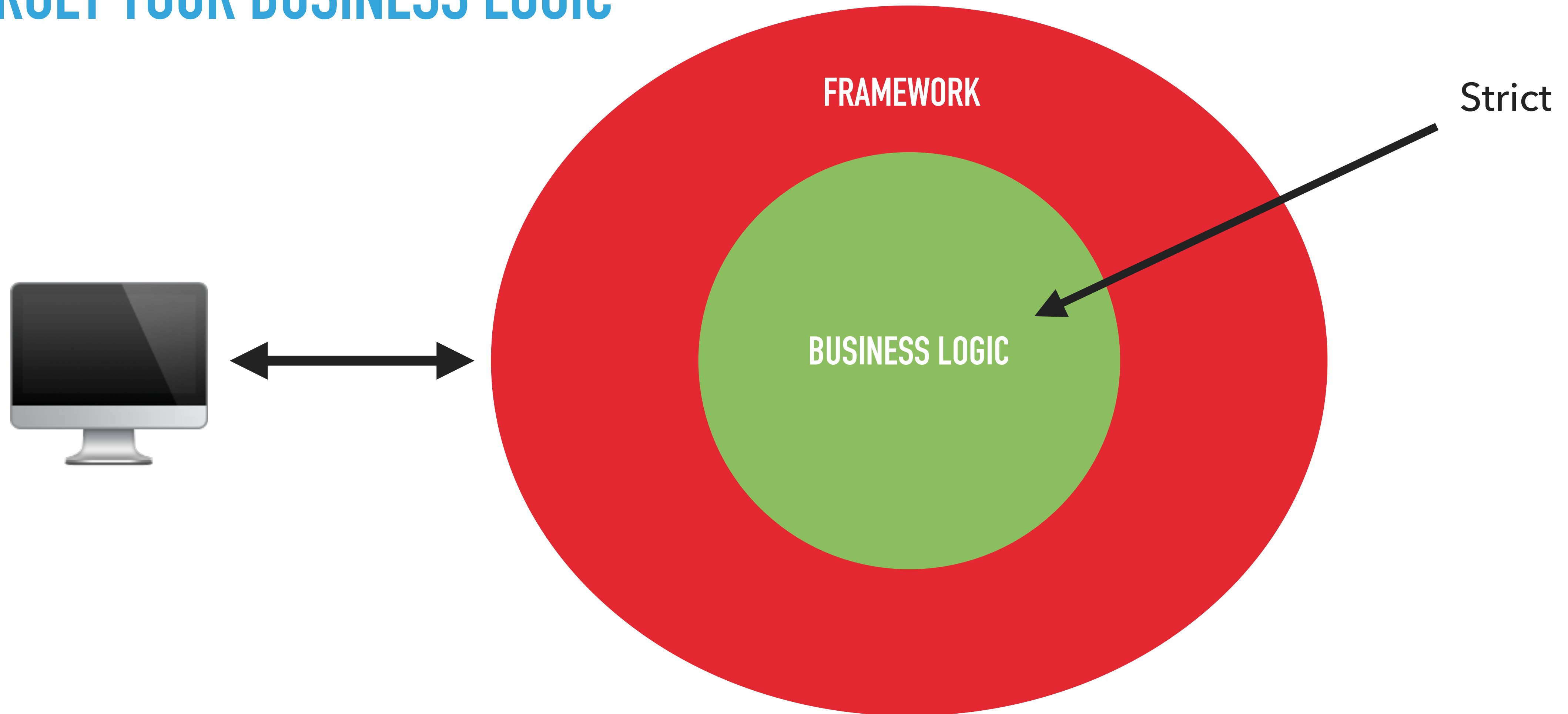
You don't really expect me to fix  
all those "bugs"?

No. Here are some tips.

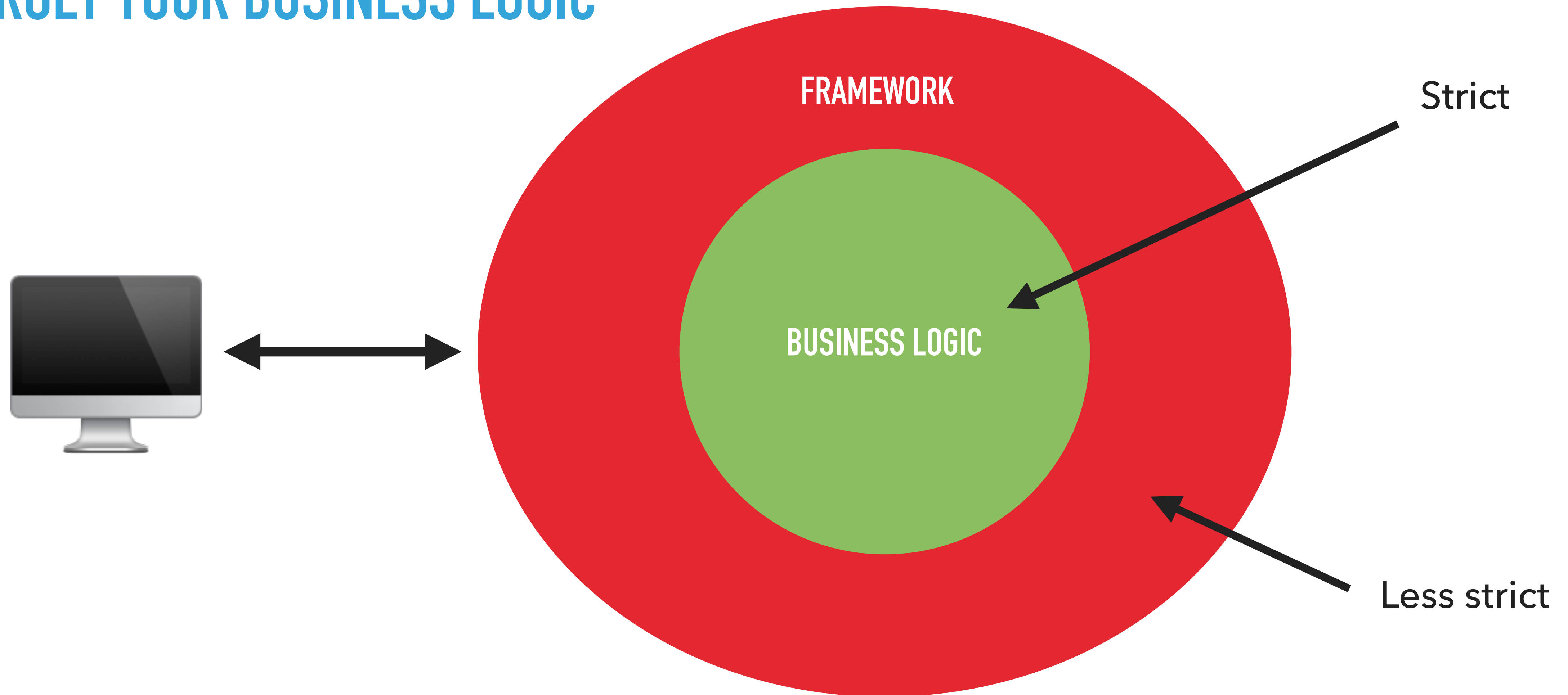
## TARGET YOUR BUSINESS LOGIC



## 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: 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);
```

## 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);
```

## 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);
```

## 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);
```

## 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);
```

## 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);
```

## 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);
```



## 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);
```

# FURTHER STATIC ANALYSIS TIPS

```
class Foo {
    public function sayHello(): void {...}
}

class DIContainer
{
    /**
     * @param string $className
     * @return mixed
     */
    public function make(string $className) {...}
}

$foo = $this->diContainer->make(Foo::class);
$foo->sayHello();
```

## FURTHER STATIC ANALYSIS TIPS

```
class Foo {  
    public function sayHello(): void {...}  
}
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @return mixed  
     */  
    public function make(string $className) {...}  
}
```

```
$foo = $this->diContainer->make(Foo::class);  
$foo->sayHello();
```

# FURTHER STATIC ANALYSIS TIPS

```
class Foo {  
    public function sayHello(): void {...}  
}
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @return mixed  
     */  
    public function make(string $className) {...}  
}
```

```
$foo = $this->diContainer->make(Foo::class);  
$foo->sayHello();
```

# FURTHER STATIC ANALYSIS TIPS

```
class Foo {  
    public function sayHello(): void {...}  
}
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @return mixed  
     */  
    public function make(string $className) {...}  
}
```

```
$foo = $this->diContainer->make(Foo::class);  
$foo->sayHello();
```

# FURTHER STATIC ANALYSIS TIPS

```
class Foo {  
    public function sayHello(): void {...}  
}
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @return mixed  
     */  
    public function make(string $className) {...}  
}
```

```
$foo = $this->diContainer->make(Foo::class);  
$foo->sayHello();
```

# FURTHER STATIC ANALYSIS TIPS

```
class Foo {  
    public function sayHello(): void {...}  
}
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @return mixed  
     */  
    public function make(string $className) {...}  
}
```

```
$foo = $this->diContainer->make(Foo::class);  
$foo->sayHello();
```

# FURTHER STATIC ANALYSIS TIPS

```
class Foo {  
    public function sayHello(): void {...}  
}
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @return mixed  
     */  
    public function make(string $className) {...}  
}
```

```
$foo = $this->diContainer->make(Foo::class);  
$foo->sayHello();
```



# FURTHER STATIC ANALYSIS TIPS

```
class Foo {  
    public function sayHello(): void {...}  
}
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @return mixed  
     */  
    public function make(string $className) {...}  
}
```

```
$foo = $this->diContainer->make(Foo::class);  
$foo->sayHello();
```

# FURTHER STATIC ANALYSIS TIPS

```
class Foo {
    public function sayHello(): void {...}
}

class DIContainer
{
    /**
     * @param string $className
     * @return mixed
     */
    public function make(string $className) {...}
}

/** @var Foo $foo */
$foo = $this->diContainer->make(Foo::class);
$foo->sayHello();
```

# FURTHER STATIC ANALYSIS TIPS

```
class Foo {
    public function sayHello(): void {...}
}

class DIContainer
{
    /**
     * @param string $className
     * @return mixed
     */
    public function make(string $className) {...}
}

/** @var Foo $foo */
$foo = $this->diContainer->make(Foo::class);
$foo->sayHello();
```

## FURTHER STATIC ANALYSIS TIPS

```
$foo = $this->diContainer->make ( '\MyApp\Foo' ) ;  
$foo->sayHello ( ) ;
```

## FURTHER STATIC ANALYSIS TIPS

```
$foo = $this->diContainer->make (' \MyApp\Foo' );  
$foo->sayHello();
```

## FURTHER STATIC ANALYSIS TIPS

```
$foo = $this->diContainer->make ( '\MyApp\Foo' ) ;  
$foo->sayHello ( ) ;
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @psalm-param class-string $className  
     * @return mixed  
     */  
    public function make(string $className) {...}  
}
```

## FURTHER STATIC ANALYSIS TIPS

```
$foo = $this->diContainer->make ( '\ MyApp\Foo' ) ;  
$foo->sayHello ( ) ;
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @psalm-param class-string $className  
     * @return mixed  
     */  
    public function make(string $className) {...}  
}
```

# FURTHER STATIC ANALYSIS TIPS

```
/** @var Foo $foo */  
$foo = $this->diContainer->make(Bar::class);  
$foo->sayHello();
```



# FURTHER STATIC ANALYSIS TIPS

```
/** @var Foo $foo */  
$foo = $this->diContainer->make(Bar::class);  
$foo->sayHello();
```

### FURTHER STATIC ANALYSIS TIPS

```
/** @var Foo $foo */  
$foo = $this->diContainer->make(Bar::class);  
$foo->sayHello();
```

# FURTHER STATIC ANALYSIS TIPS

```
/** @var Foo $foo */  
$foo = $this->diContainer->make(Bar::class);  
$foo->sayHello();
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @psalm-param class-string $className  
     * @template T  
     * @template-typeof T $className  
     * @psalm-return T  
     */  
    public function make(string $className) {...}  
}
```

# FURTHER STATIC ANALYSIS TIPS

```
/** @var Foo $foo */  
$foo = $this->diContainer->make(Bar::class);  
$foo->sayHello();
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @psalm-param class-string $className  
     * @template T  
     * @template-typeof T $className  
     * @psalm-return T  
     */  
    public function make(string $className) {...}  
}
```

# FURTHER STATIC ANALYSIS TIPS

```
/** @var Foo $foo */  
$foo = $this->diContainer->make(Bar::class);  
$foo->sayHello();
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @psalm-param class-string $className  
     * @template T  
     * @template-typeof T $className  
     * @psalm-return T  
     */  
    public function make(string $className) {...}  
}
```

# FURTHER STATIC ANALYSIS TIPS

```
/** @var Foo $foo */  
$foo = $this->diContainer->make(Bar::class);  
$foo->sayHello();
```

```
class DIContainer  
{  
    /**  
     * @param string $className  
     * @psalm-param class-string $className  
     * @template T  
     * @template-typeof T $className  
     * @psalm-return T  
     */  
    public function make(string $className) {...}  
}
```

## FURTHER STATIC ANALYSIS TIPS

```
class LoginCommand
{
    public function __construct(...) {...}

    public function execute(): void {...}

    public function getAccessToken(): string {...}
}
```

## FURTHER STATIC ANALYSIS TIPS

```
class LoginCommand
{
    public function __construct(...) {...}

    public function execute(): void {...}

    public function getAccessToken(): string {...}
}
```

```
$login = new LoginCommand();
$login->getAccessToken();
```



## FURTHER STATIC ANALYSIS TIPS

```
/**
 * @var string
 */
private $accessToken;

public function getAccessToken(): string
{

    return $this->accessToken;
}
```

## FURTHER STATIC ANALYSIS TIPS

```
/**  
 * @var string  
 */  
private $accessToken;
```

```
public function getAccessToken(): string  
{  
  
    return $this->accessToken;  
}
```

## FURTHER STATIC ANALYSIS TIPS

```
/**
 * @var string|null
 */
private $accessToken;

public function getAccessToken(): string
{

    return $this->accessToken;
}
```

## FURTHER STATIC ANALYSIS TIPS

```
/**  
 * @var string|null  
 */  
private $accessToken;  
  
public function getAccessToken(): string  
{  
  
    return $this->accessToken;  
}
```

## FURTHER STATIC ANALYSIS TIPS

```
/**  
 * @var string|null  
 */  
private $accessToken;
```

```
public function getAccessToken(): string  
{  
  
    return $this->accessToken;  
}
```

# FURTHER STATIC ANALYSIS TIPS

```
/**  
 * @var string|null  
 */  
private $accessToken;
```

```
public function getAccessToken(): string  
{  
  
    return $this->accessToken;  
}
```

# FURTHER STATIC ANALYSIS TIPS

```
/**
 * @var string|null
 */
private $accessToken;

public function getAccessToken(): string
{
    if ($this->accessToken === null) {
        throw new LogicException(... message ...);
    }
    return $this->accessToken;
}
```

# FURTHER STATIC ANALYSIS TIPS

```
/**
 * @var string|null
 */
private $accessToken;

public function getAccessToken(): string
{
    if ($this->accessToken === null) {
        throw new LogicException(... message ...);
    }
    return $this->accessToken;
}
```



## FURTHER STATIC ANALYSIS TIPS

```
/**
 * @var string|null
 */
private $accessToken;

public function getAccessToken(): string
{
    Assert::notNull($this->accessToken, ...message...);

    return $this->accessToken;
}
```

## FURTHER STATIC ANALYSIS TIPS

```
/**  
 * @var string|null  
 */  
private $accessToken;
```

```
public function getAccessToken(): string  
{  
    Assert::notNull($this->accessToken, ...message...);  
    return $this->accessToken;  
}
```

# FURTHER STATIC ANALYSIS TIPS

```
class Assert
{
    /**
     * @param mixed|null $expression
     * @param string $message
     */
    public static function notNull($expression, string $message): void
    {
        if ($expression === null) {
            throw new LogicException($message);
        }
    }
}
```

# FURTHER STATIC ANALYSIS TIPS

```
class Assert
{
    public static function notNull($expression, string $message): void
    {
        self::assertTrue($expression !== null, $message);
    }

    ... other assertions ...

    public static function assertTrue($expression, string $message): void
    {
        if ($expression !== true) {
            throw new LogicException($message);
        }
    }
}
```

### FURTHER STATIC ANALYSIS TIPS

```
class Assert
```

```
{
```

```
    public static function notNull($expression, string $message): void
```

```
    {
```

```
        self::assertTrue($expression !== null, $message);
```

```
    }
```

```
    ... other assertions ...
```

```
    public static function assertTrue($expression, string $message): void
```

```
    {
```

```
        if ($expression !== true) {
```

```
            throw new LogicException($message);
```

```
        }
```

```
    }
```

```
}
```

### FURTHER STATIC ANALYSIS TIPS

```
class Assert
{
    public static function notNull($expression, string $message): void
    {
        self::assertTrue($expression !== null, $message);
    }

    ... other assertions ...

    public static function assertTrue($expression, string $message): void
    {
        if ($expression !== true) {
            throw new LogicException($message);
        }
    }
}
```

## FURTHER STATIC ANALYSIS TIPS

```
class Assert
{

    public static function notNull($expression, string $message): void
    {
        self::assertTrue($expression !== null, $message);
    }

}
```

# FURTHER STATIC ANALYSIS TIPS

```
class Assert
{
    /**
     * @psalm-assert !null $expression
     */
    public static function notNull($expression, string $message): void
    {
        self::assertTrue($expression !== null, $message);
    }
}
```



# FURTHER STATIC ANALYSIS TIPS

```
class Assert
{
    /**
     * @psalm-assert !null $expression
     */
    public static function notNull($expression, string $message): void
    {
        self::assertTrue($expression !== null, $message);
    }
}
```

## FURTHER STATIC ANALYSIS TIPS

What about 3rd  
party libraries?

## FURTHER STATIC ANALYSIS TIPS

## FURTHER STATIC ANALYSIS TIPS

Stubs/Assert.php

# FURTHER STATIC ANALYSIS TIPS

### Stubs/Assert.php

```
namespace Webmozart\Assert;

class Assert
{
    /**
     * @psalm-assert !null $value
     */
    public static function notNull($value, $message='') {}

    ... other functions ...
}
"
```

# FURTHER STATIC ANALYSIS TIPS

### Stubs/Assert.php

```
namespace Webmozart\Assert;
```

```
class Assert  
{
```

```
    /**
```

```
     * @psalm-assert !null $value
```

```
     */
```

```
    public static function notNull($value, $message='') {}
```

```
    ... other functions ...
```

```
    ") {}
```

```
}
```

## FURTHER STATIC ANALYSIS TIPS

```
<psalm ...>
```

```
... other config ...
```

```
<stubs>
```

```
  <file name="Stubs/Assert.php" />
```

```
  ... other stub files ...
```

```
</stubs>
```

```
<psalm>
```

## FURTHER STATIC ANALYSIS TIPS

```
<psalm ...>
```

```
... other config ...
```

```
<stubs>
```

```
<file name="Stubs/Assert.php" />
```

```
... other stub files ...
```

```
</stubs>
```

```
<psalm>
```

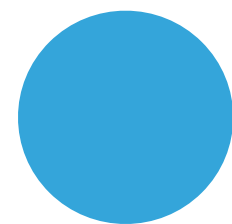


## LEARN FROM MISTAKES AND DON'T BE SLOPPY

- ▶ Learn from issues raised
- ▶ Type hint everything
- ▶ Create / use plugins / stubs to give extra information to static analysis tools

## CHAPTER 6:

## CHAPTER 6: BASELINE STATIC ANALYSIS RESULTS



# CHAPTER 6: BASELINE STATIC ANALYSIS RESULTS

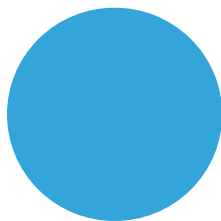
Problem

Problem

Problem

Problem

Problem



# CHAPTER 6: BASELINE STATIC ANALYSIS RESULTS

Problem

Problem

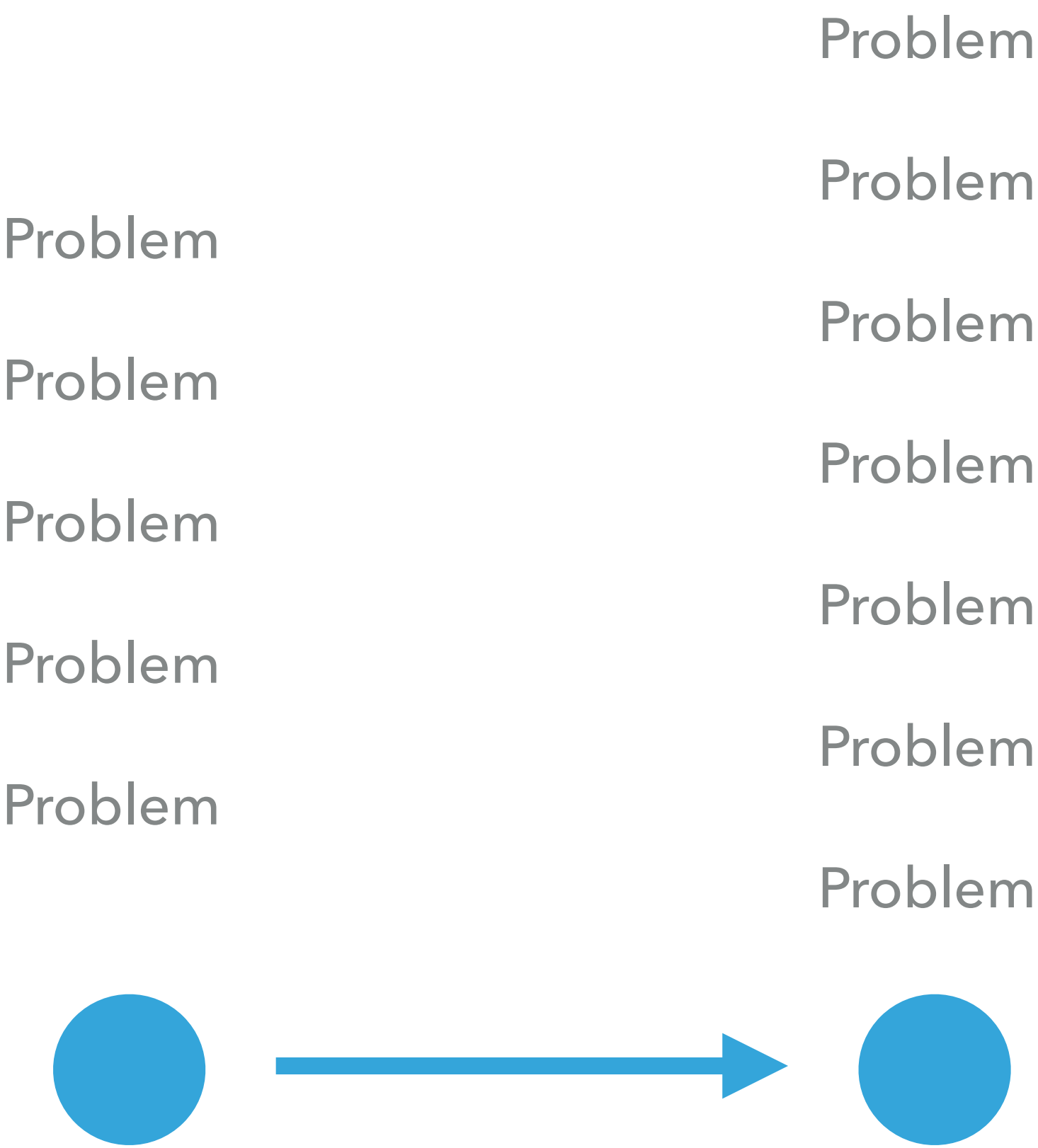
Problem

Problem

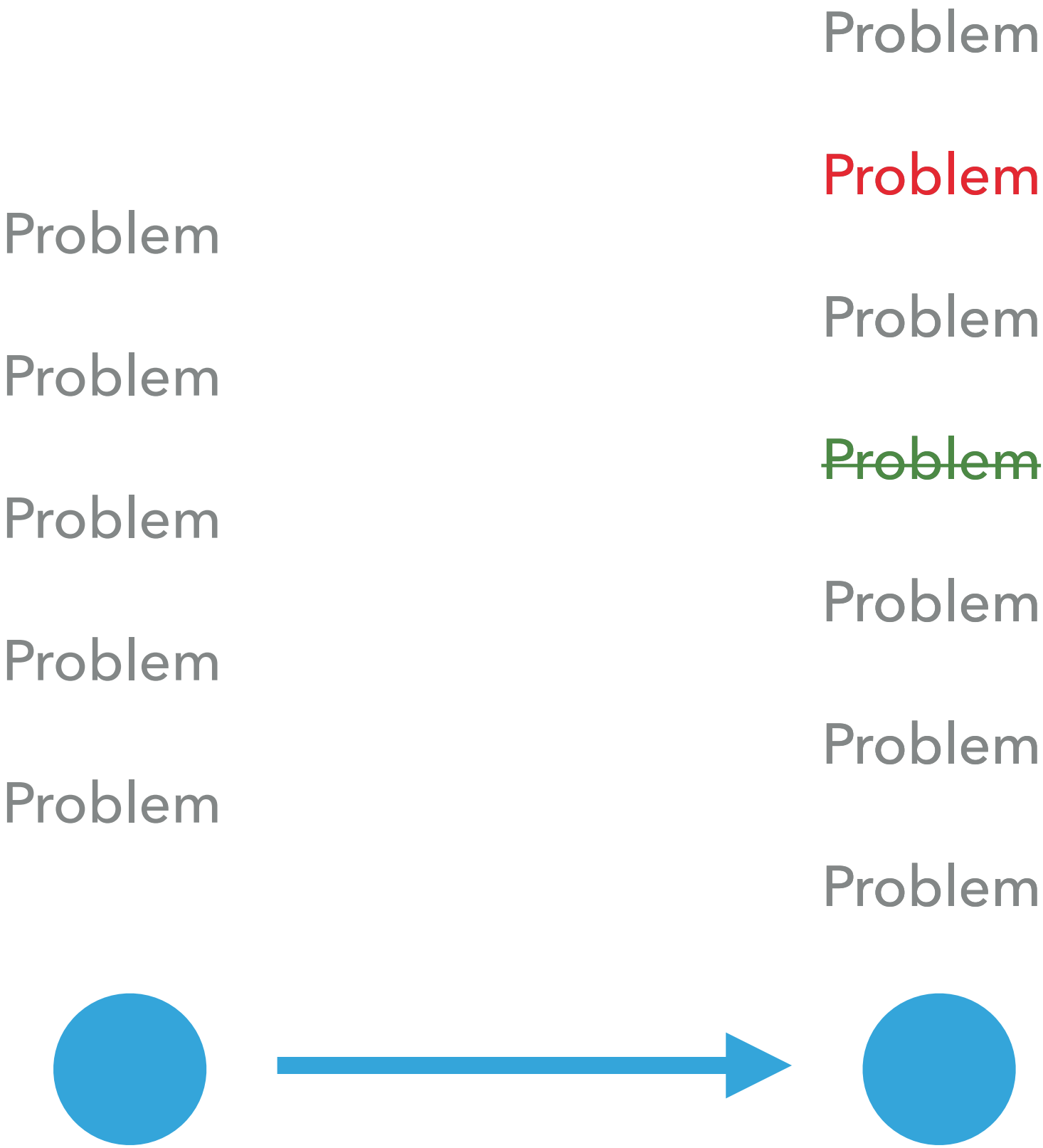
Problem



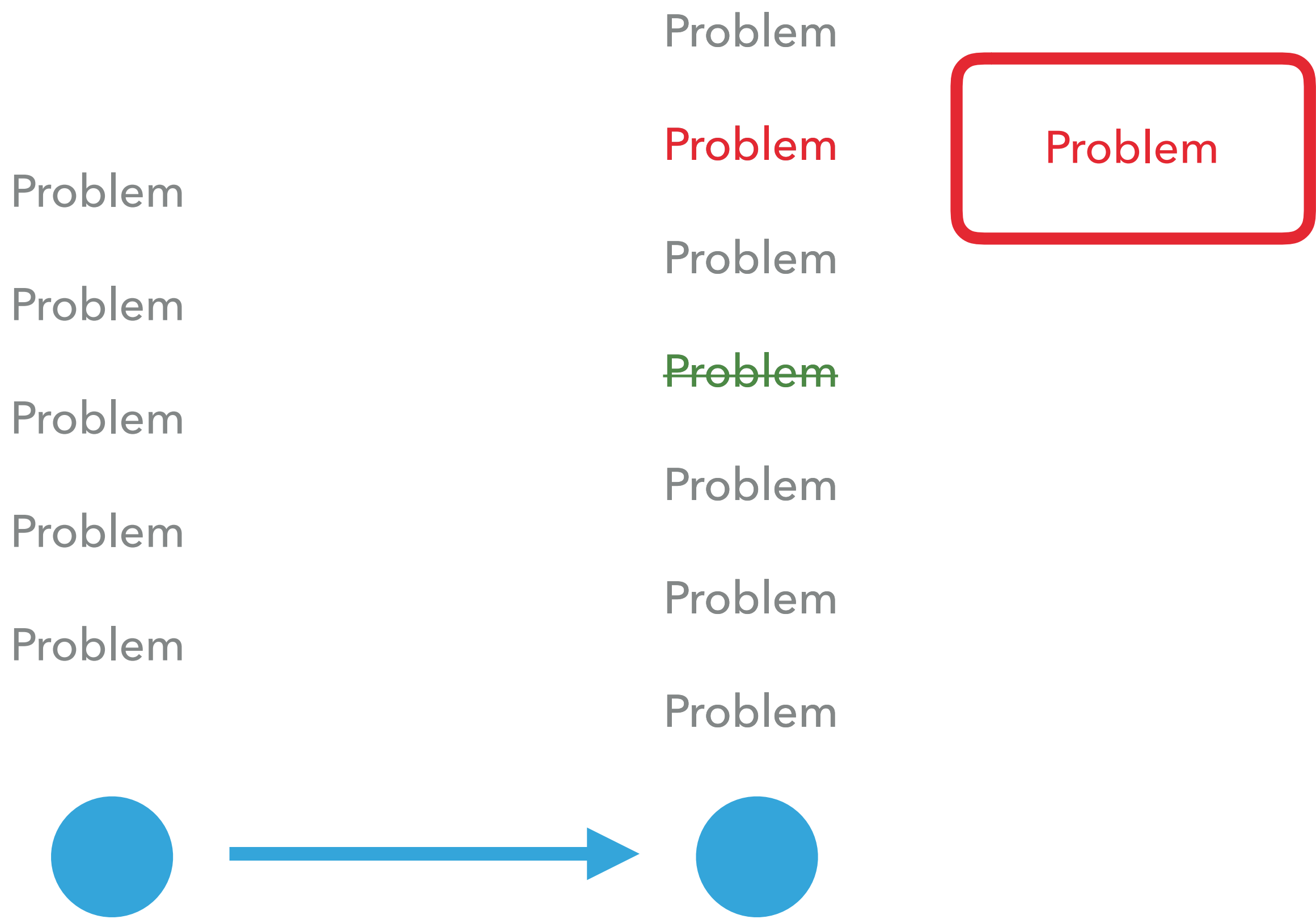
# CHAPTER 6: BASELINE STATIC ANALYSIS RESULTS



# CHAPTER 6: BASELINE STATIC ANALYSIS RESULTS



# CHAPTER 6: 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
```

```
> sarb create-baseline ... args ...
```

```
Baseline created with 328 problems.
```

```
>
```

## SARB: REMOVE BASELINE FROM RESULTS

```
# Run Psalm on the updated code
```

```
> sarb remove-baseline-results ... args ...
```

```
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

> `sarb remove-baseline-results ... args ...`

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: BASELINE

```
class Person  
  
{  
  
    ... Some code ...  
  
    public function foo()  
    {  
        ... some code ...  
        return $bar  
    }  
}
```

## SARB BEHIND THE SCENES: BASELINE

```
class Person
{
    ... Some code ...

    public function foo()
    {
        ... some code ...
        return $bar
    }
}
```

Line 93: InvalidNullableReturnType

## SARB BEHIND THE SCENES: AFTER CODING

```
class Person  
  
{  
  
    ... Some code ...  
  
    public function foo()  
    {  
        ... some code ...  
        return $bar  
    }  
}
```



# SARB BEHIND THE SCENES: AFTER CODING

```
class Person Employee
{

... Some code ...

public function foo()
{
... some code ...
    return $bar
}
```

## SARB BEHIND THE SCENES: AFTER CODING

```
class Person Employee
```

```
{
```

```
... some code ...
```

Remove 20 lines of code

```
public function foo()
```

```
{
```

```
... some code ...
```

```
    return $bar
```

```
}
```

## SARB BEHIND THE SCENES: AFTER CODING

```
class Person Employee
```

```
{
```

```
... some code ...
```

```
public function foo()
```

```
{
```

```
... some code ...
```

```
return $bar
```

```
}
```

Remove 20 lines of code

Line 73: InvalidNullableReturnType

# 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. This problem was in the baseline. Don't report as new issue.

# 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 SARB baseline



# STATIC ANALYSIS WITH SARB

- ▶ Run static analysis tool
- ▶ Fix all bugs you decide need fixing
- ▶ Run static analysis tool again
- ▶ Generate SARB baseline
- ▶ Repeat forever:
  - ▶ Write code
  - ▶ Run analysis
  - ▶ Remove baseline results from latest analysis
  - ▶ Fix newly introduced bugs

# STATIC ANALYSIS WITH SARB

- ▶ Run static analysis tool
- ▶ Fix all bugs you decide need fixing
- ▶ Run static analysis tool again
- ▶ Generate SARB baseline
- ▶ Repeat forever:
  - ▶ Write code
  - ▶ Run analysis
  - ▶ Remove baseline results from latest analysis
  - ▶ Fix newly introduced bugs



# WHAT AN ADVENTURE IT HAS BEEN...

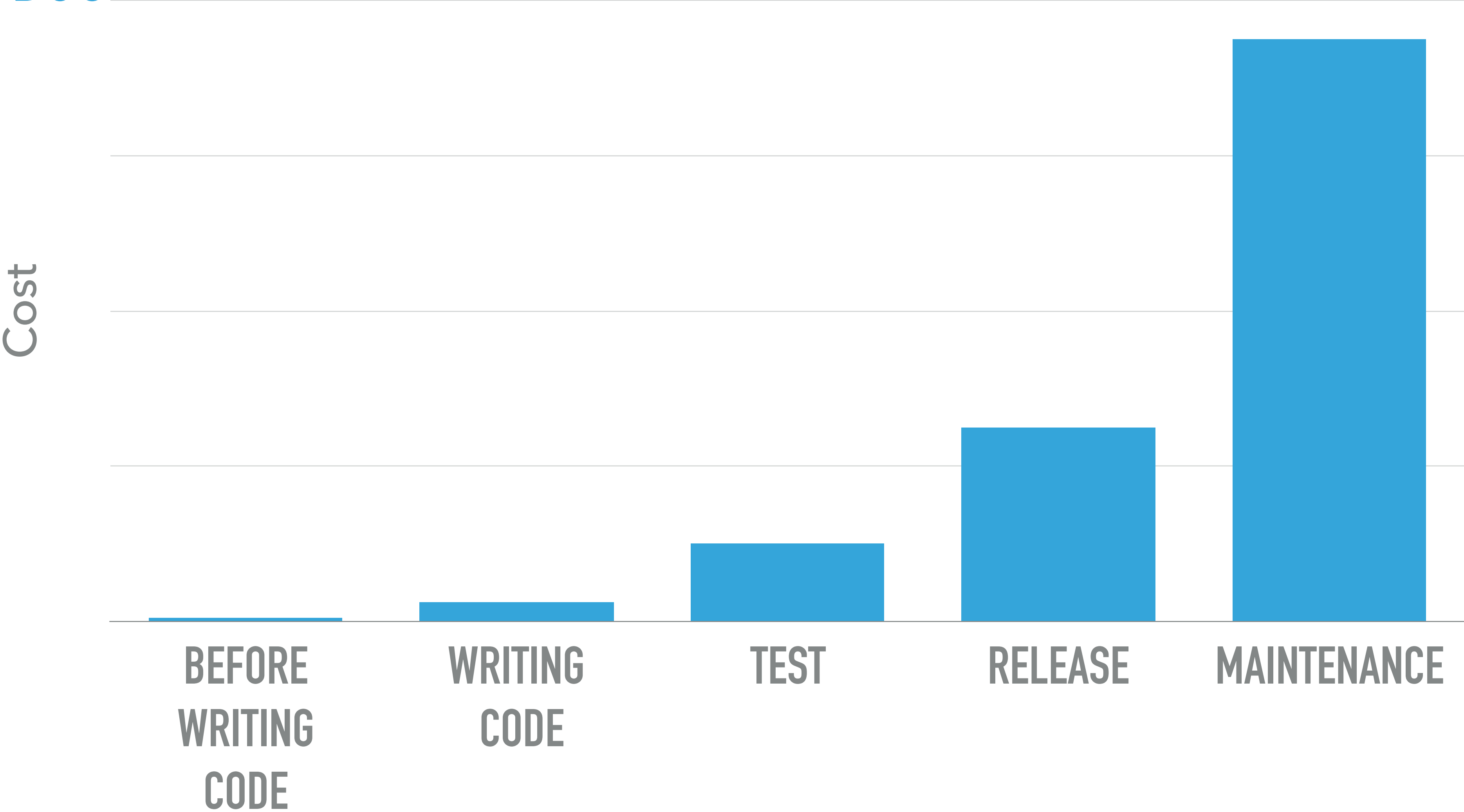
WHAT AN ADVENTURE IT HAS BEEN...

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

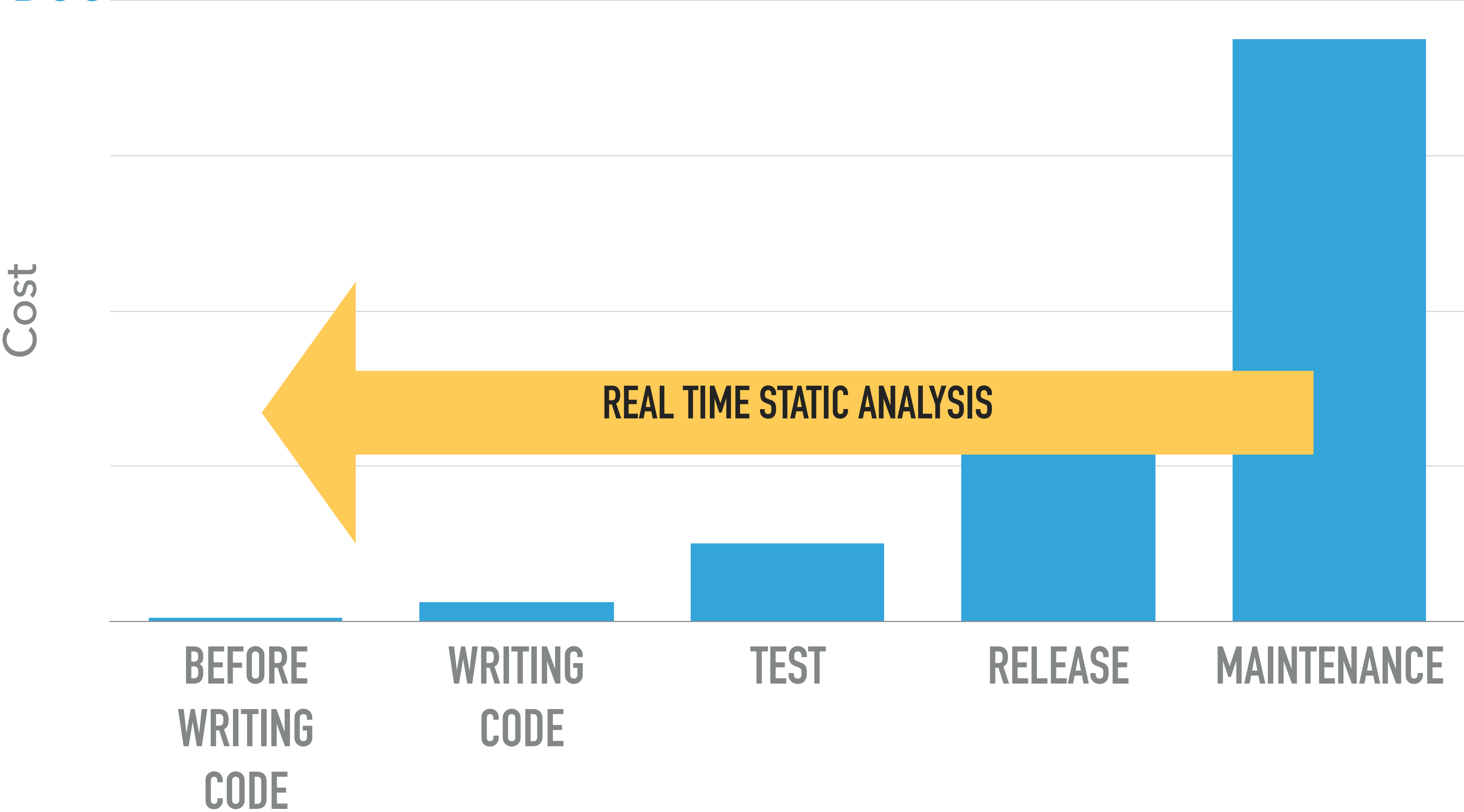
Static analysis tells you that your code is incorrect.

Tests tell you a particular scenario is working correctly.

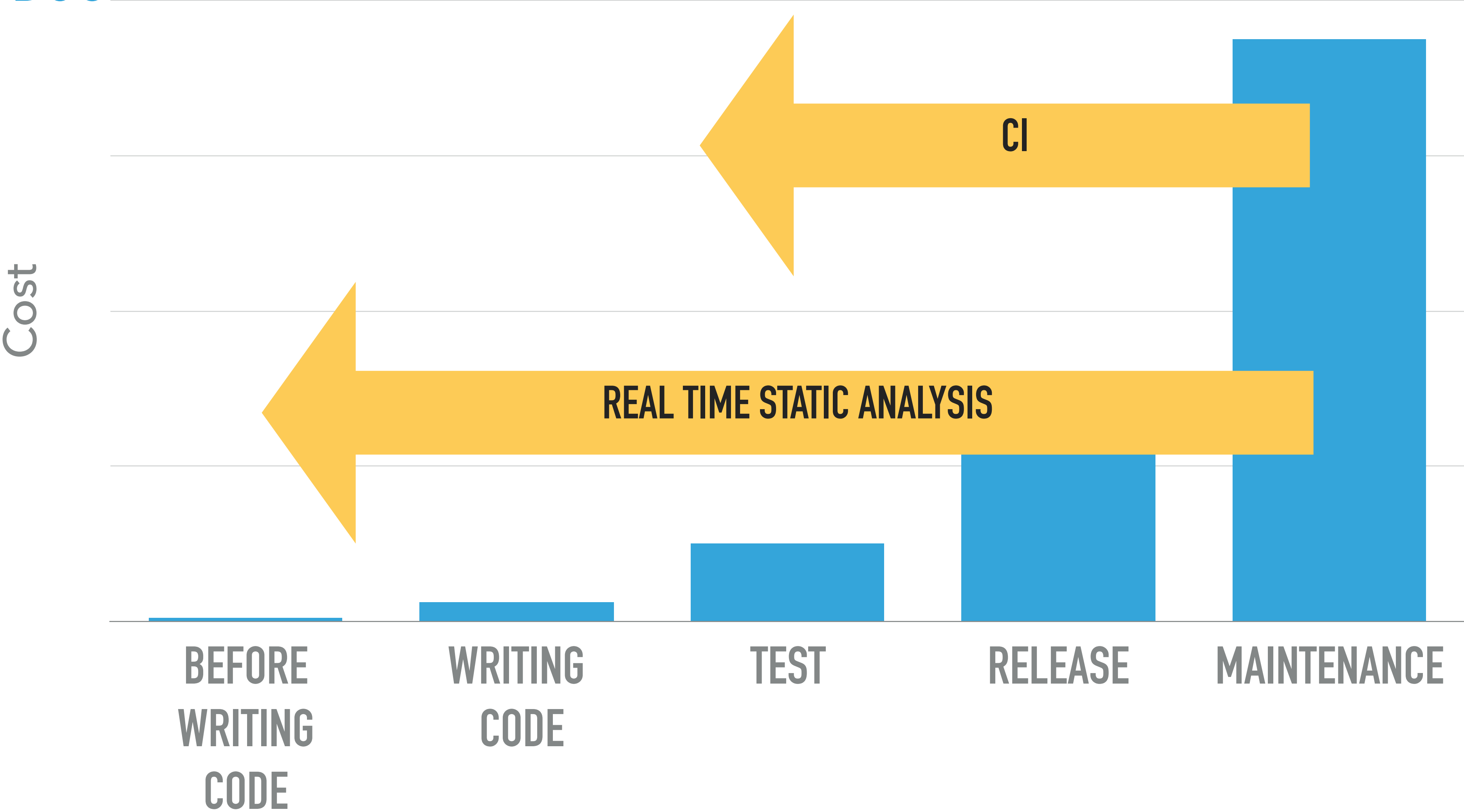
COST OF A BUG



# COST OF A BUG



# COST OF A BUG





### CI TOOLSET

- ▶ Composer validate: `composer validate --strict`
- ▶ Parallel lint: `jakub-ondarka/php-parallel-lint`
- ▶ PHP CS fixer: `friendsofsymfony/php-cs-fixer`
- ▶ Var dump checker: `jakub-ondarka/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

```
1 <?php
2
3 function foo(string $s) : void {
4     return "bar";
5 }
6
7 $a = ["hello", 5];
8 foo($a[1]);
9 foo();
10
11 if (rand(0, 1)) $b = 5;
12 echo $b;
13
14 $c = rand(0, 5);
15 if ($c) {} elseif ($c) {}
16
```

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'

[↗ Shrink](#)

[🔗 Get link](#)

# SUMMARY

---

**THANK YOU FOR  
LISTENING**

## 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/>