Is what you’ve coded what you mean?

Dave Liddament (@daveliddament)

A confession….

First let’s talk about bugs….

Question 1:
Who puts bugs in their code?

Question 2:
When is the best time to find a bug?

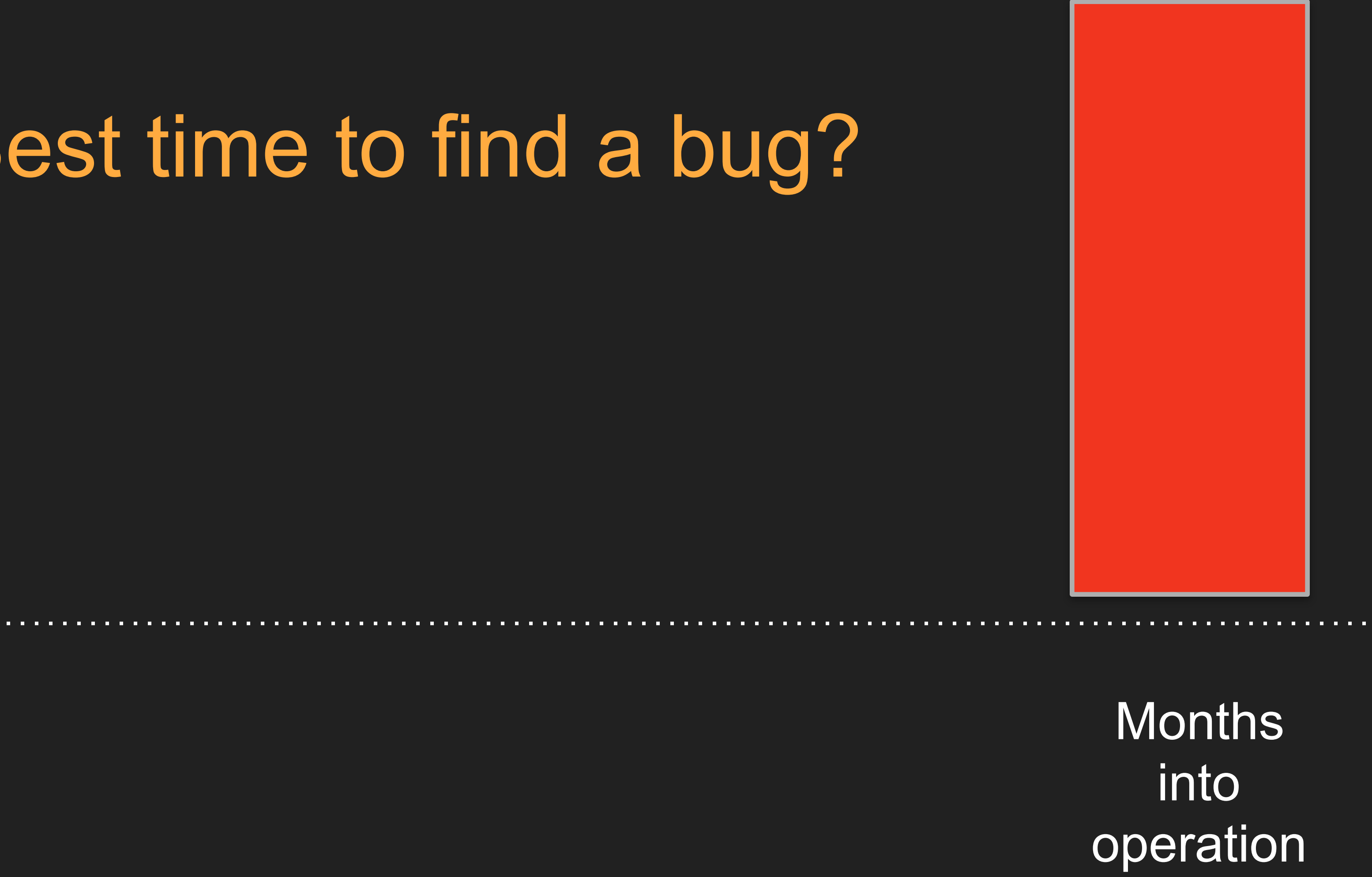
Best time to find a bug?

Best time to find a bug?

Months
into
operation

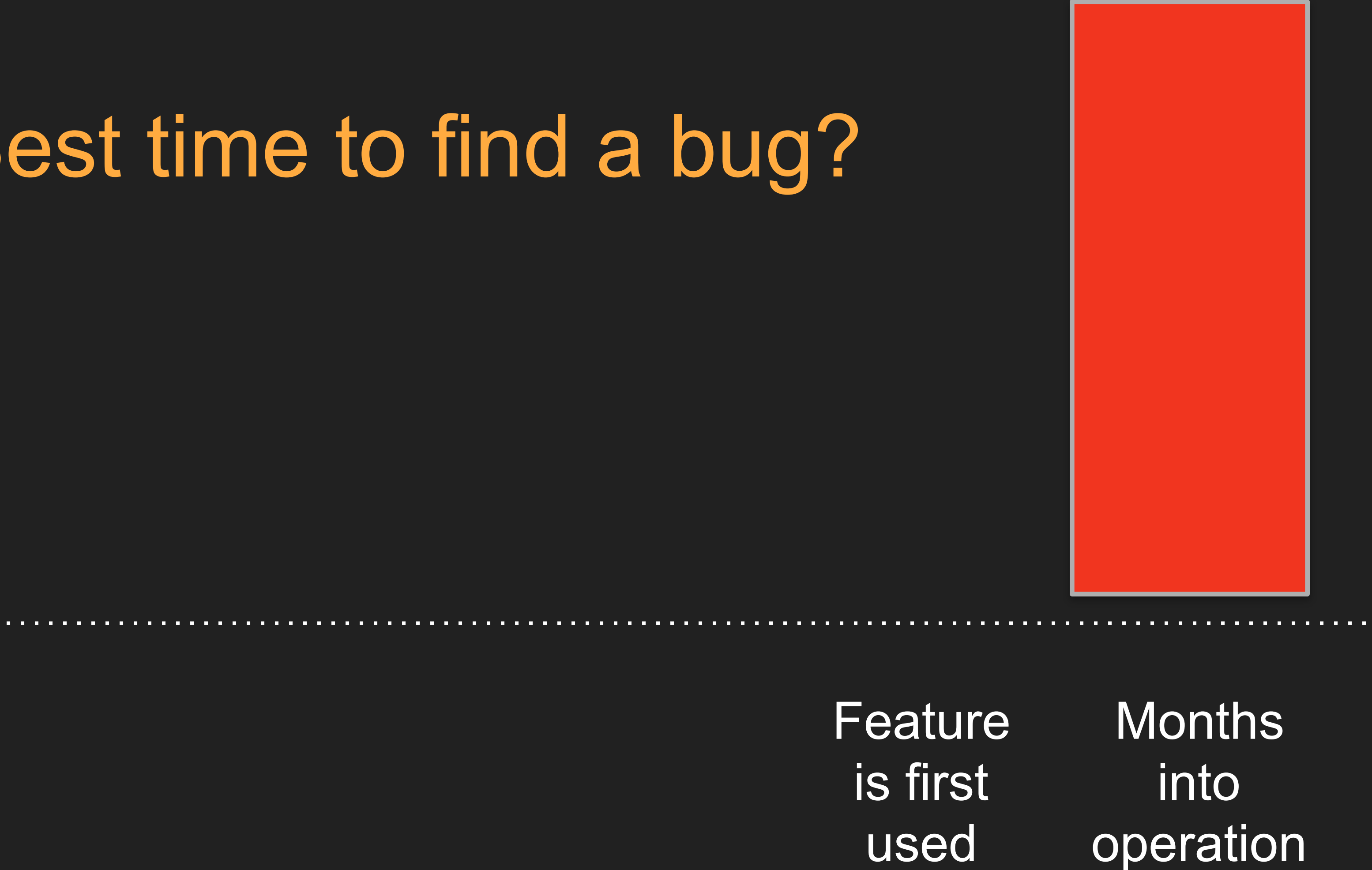
@DaveLiddament

Best time to find a bug?

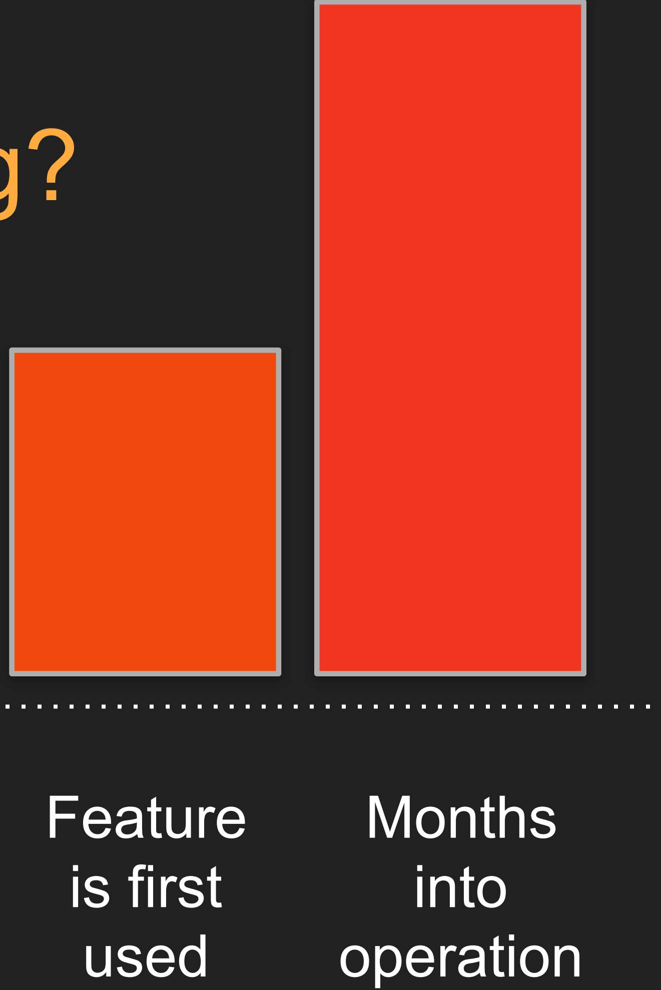


@DaveLiddament

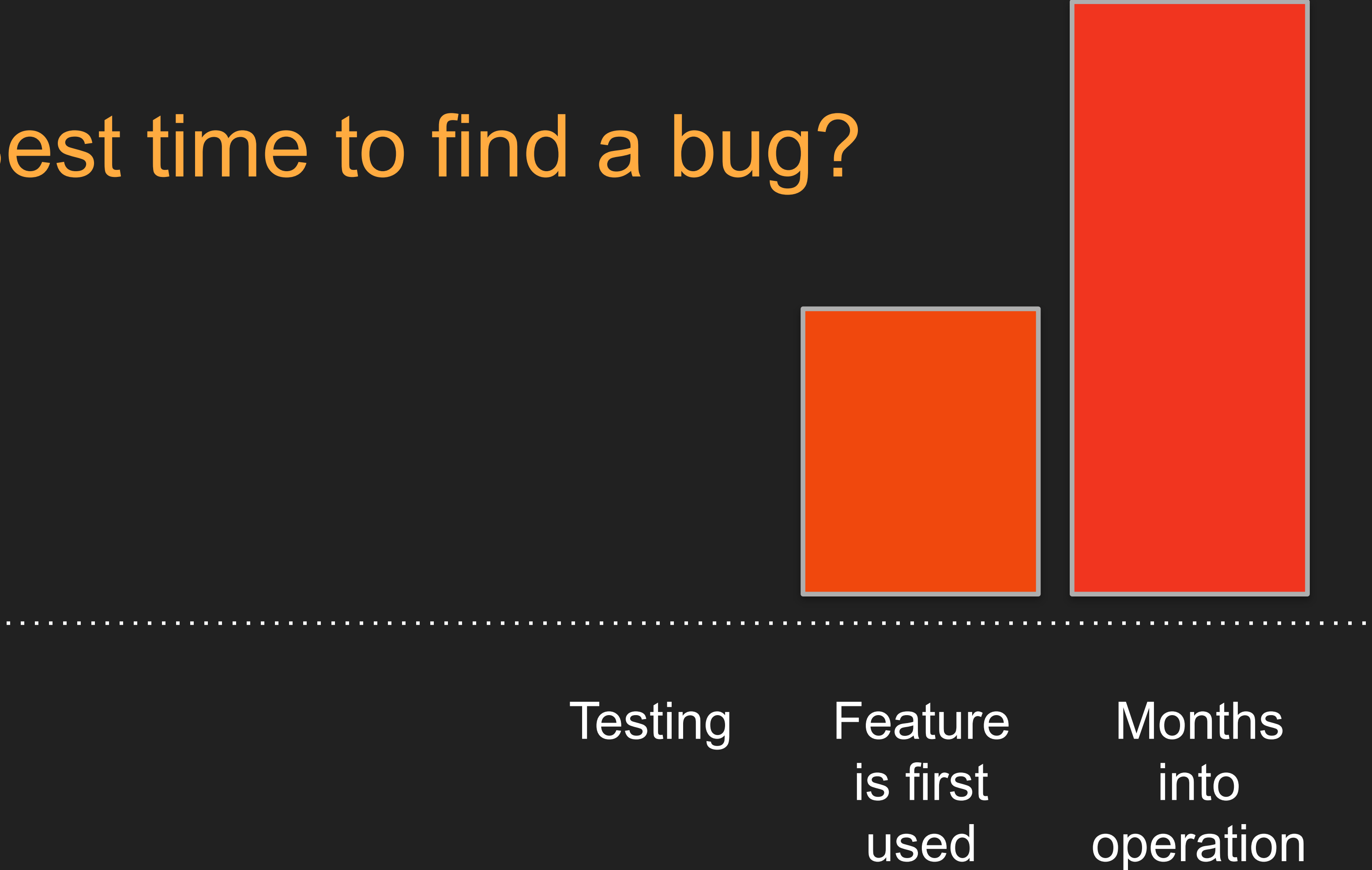
Best time to find a bug?



Best time to find a bug?

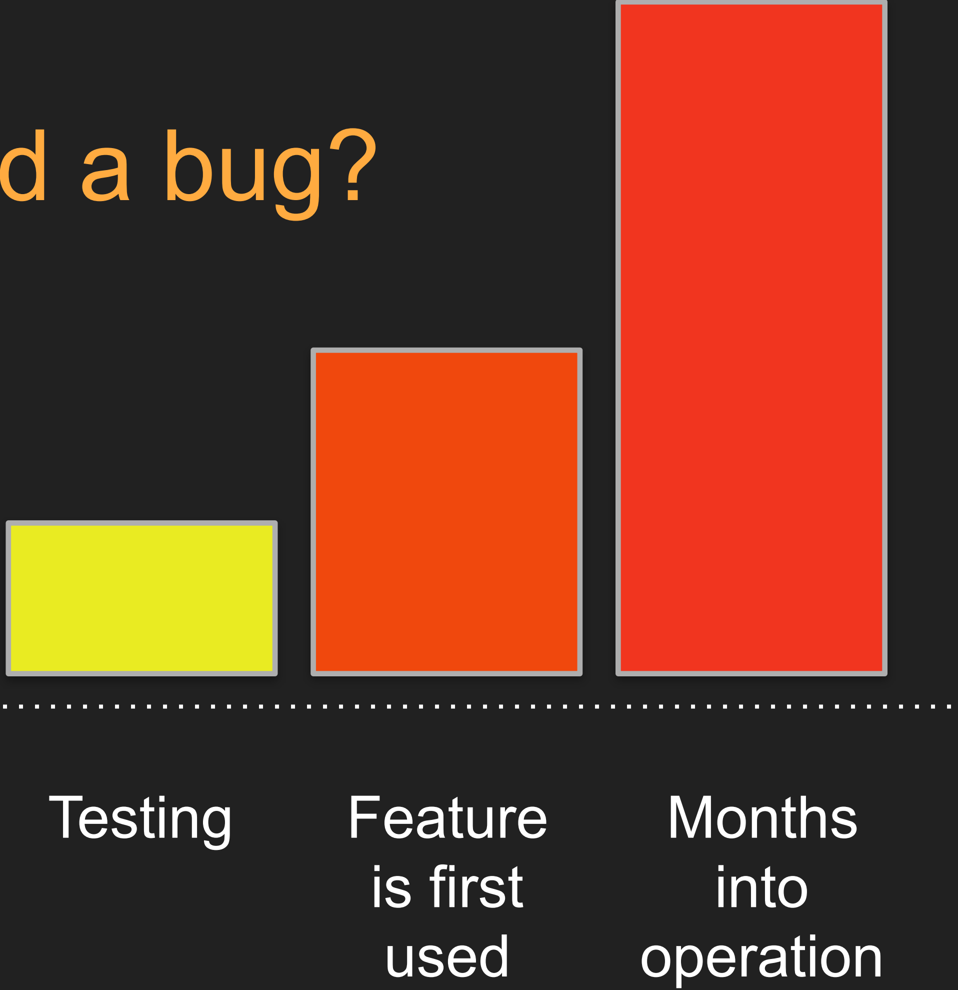


Best time to find a bug?



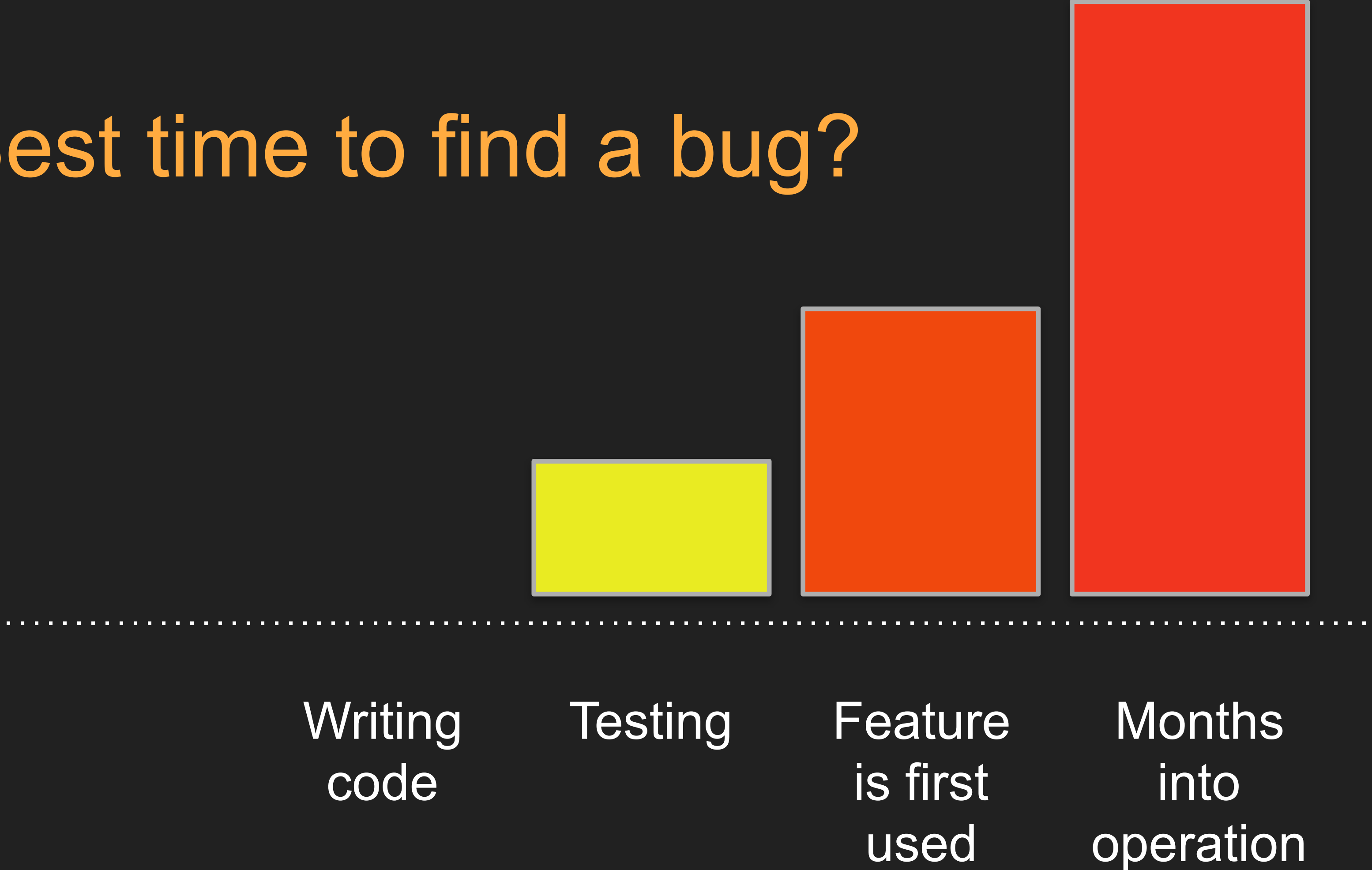
@DaveLiddament

Best time to find a bug?



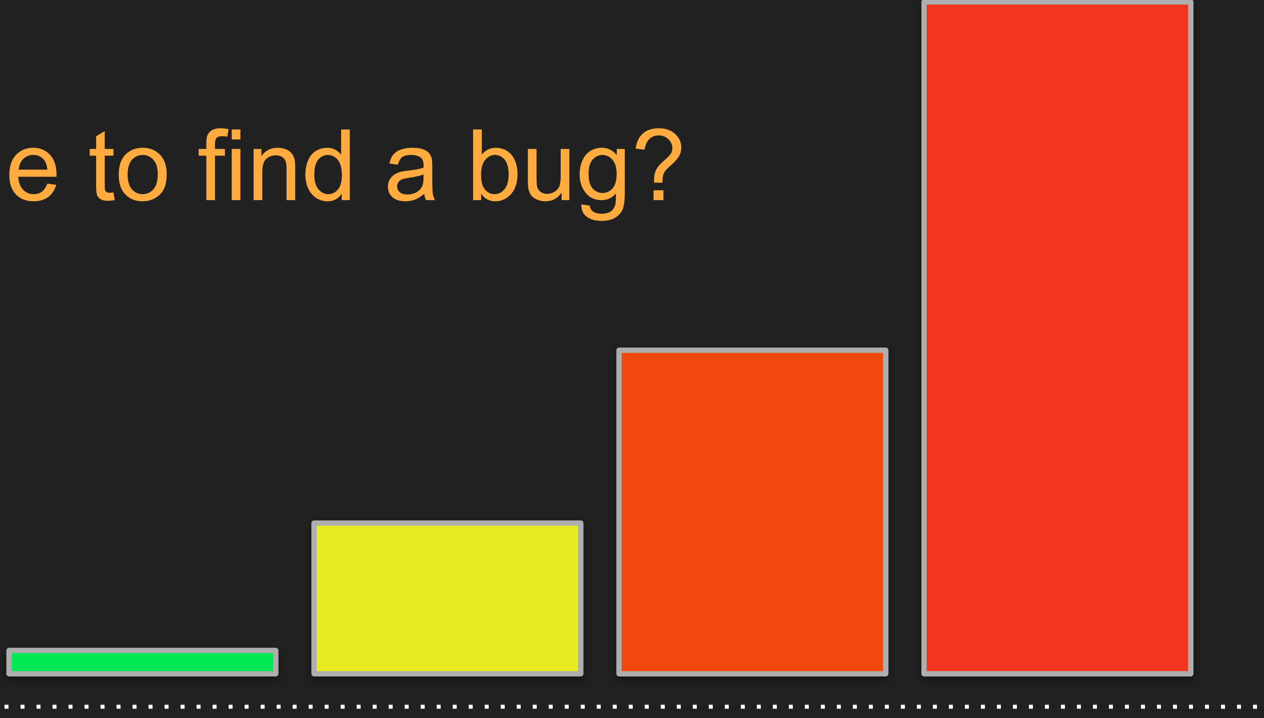
@DaveLiddament

Best time to find a bug?

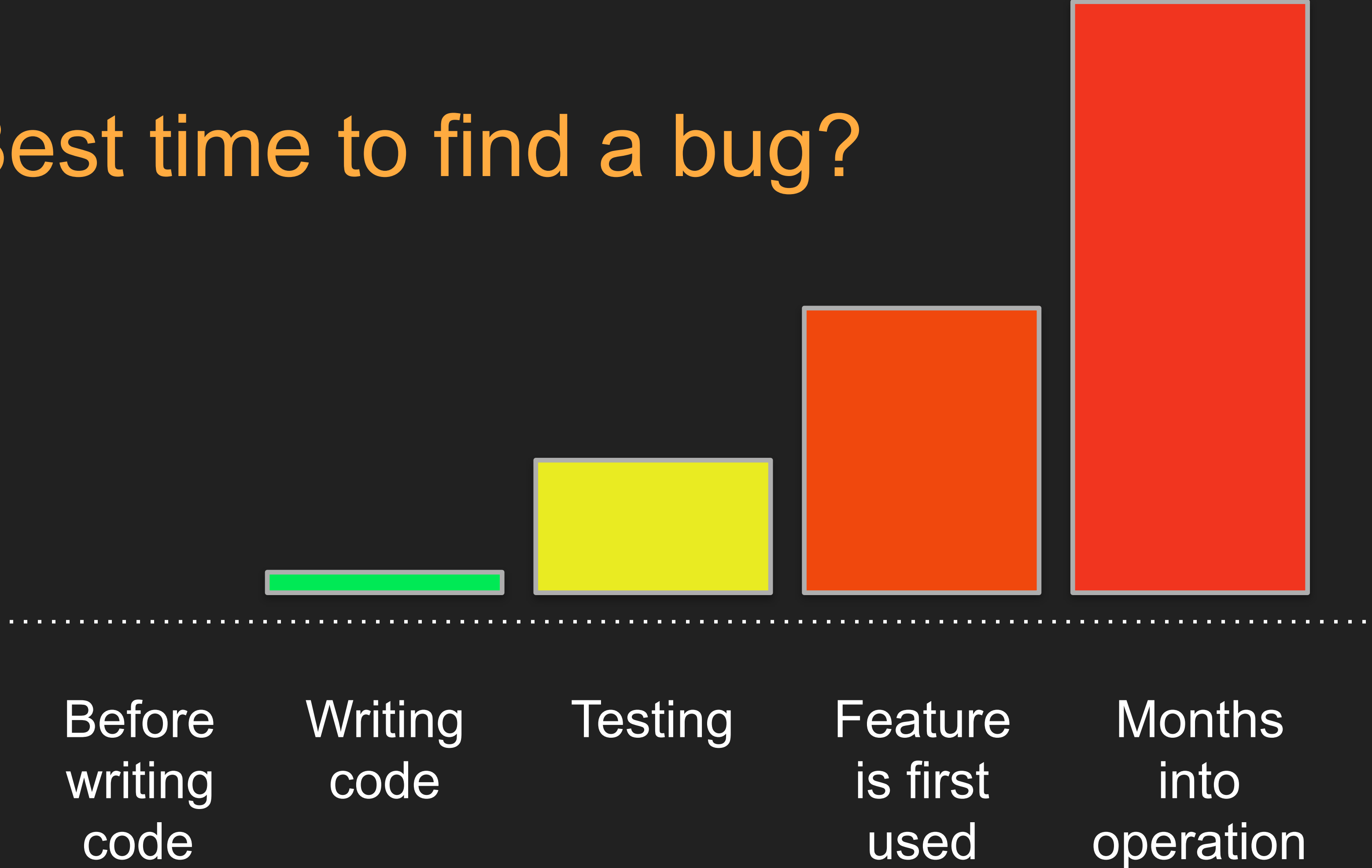


@DaveLiddament

Best time to find a bug?



Best time to find a bug?

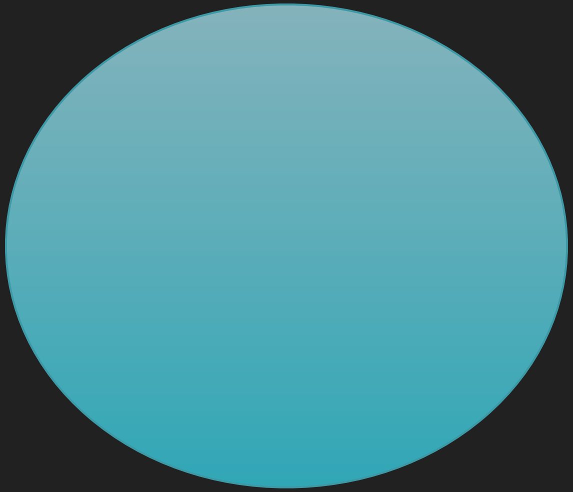


@DaveLiddament

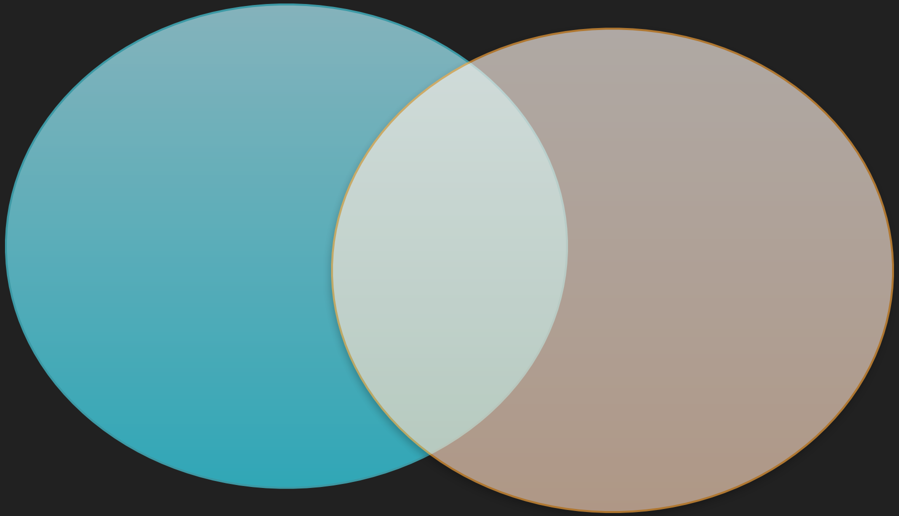
Why do bugs happen?

Why do bugs happen?

What the
code
should do



Why do bugs happen?

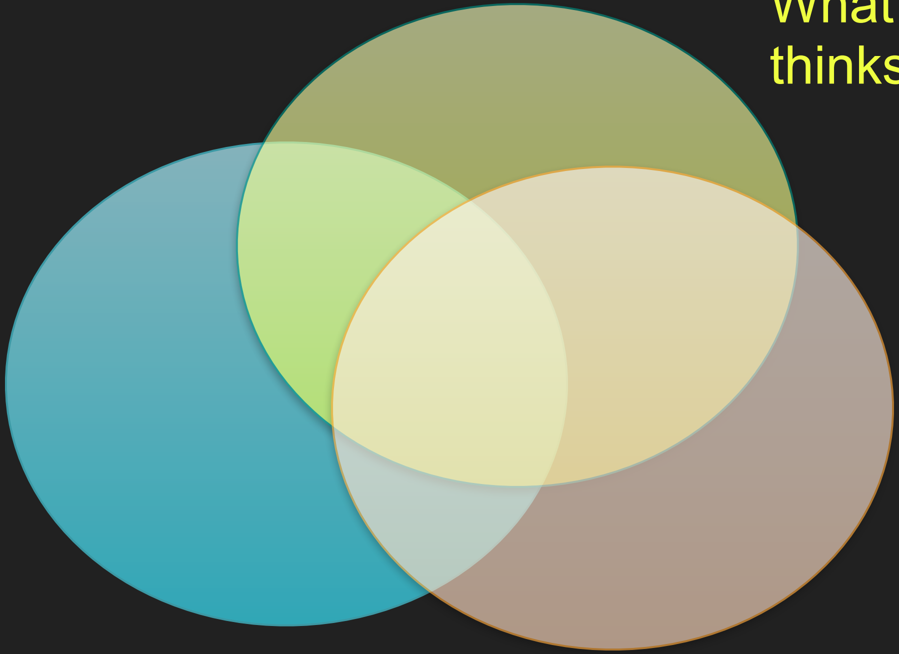


What the code should do

What the code actually does

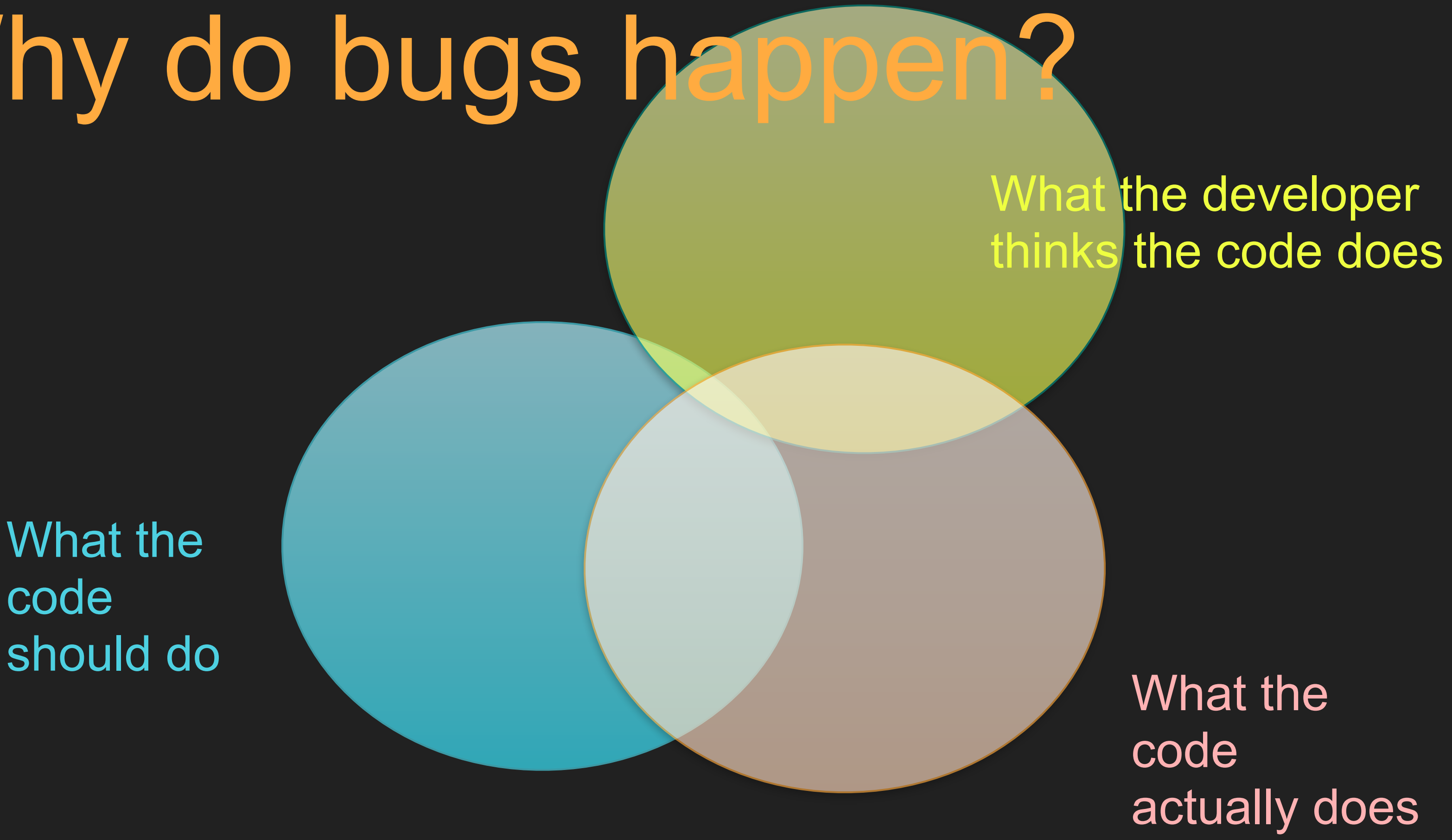
@DaveLiddament

Why do bugs happen?


What the developer thinks the code does
What the code should do
What the code actually does

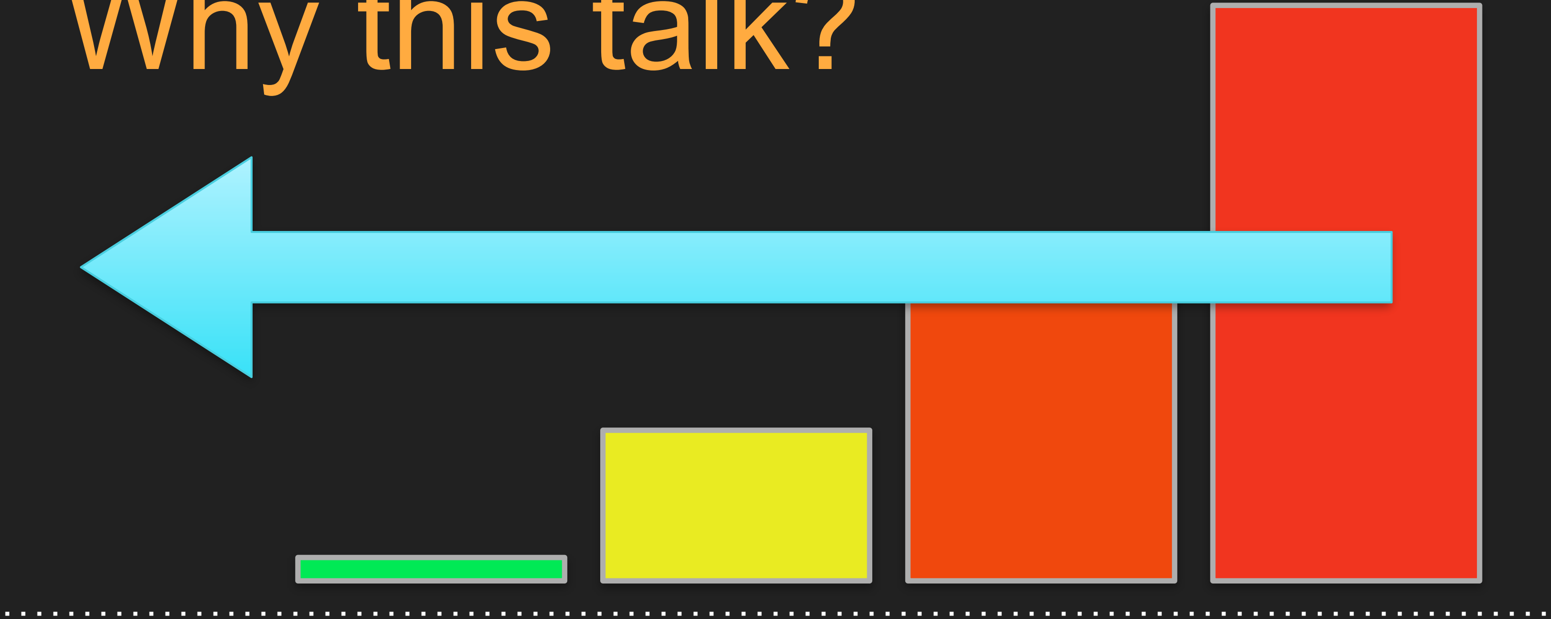
@DaveLiddament

Why do bugs happen?


What the developer thinks the code does
What the code should do
What the code actually does

@DaveLiddament

Why this talk?



Before writing code

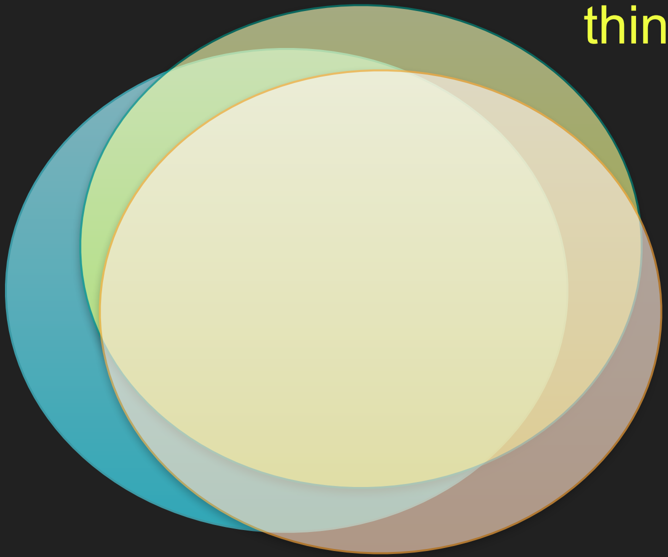
Writing code

Testing

Feature is first used

Months into operation

How we reduce cost of bugs



What the developer thinks the code does
What the code should do
What the code actually does

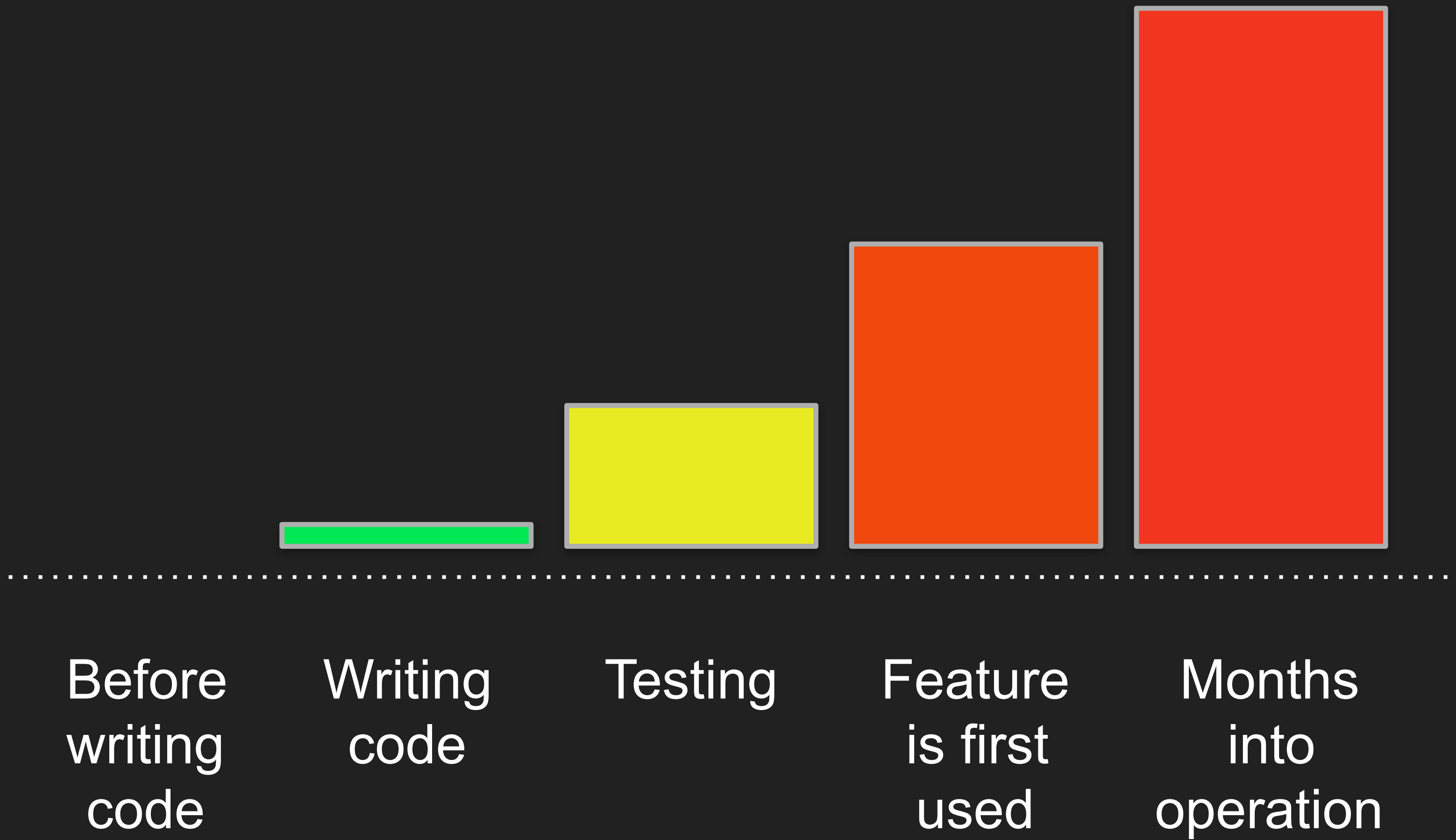
Dave Liddament

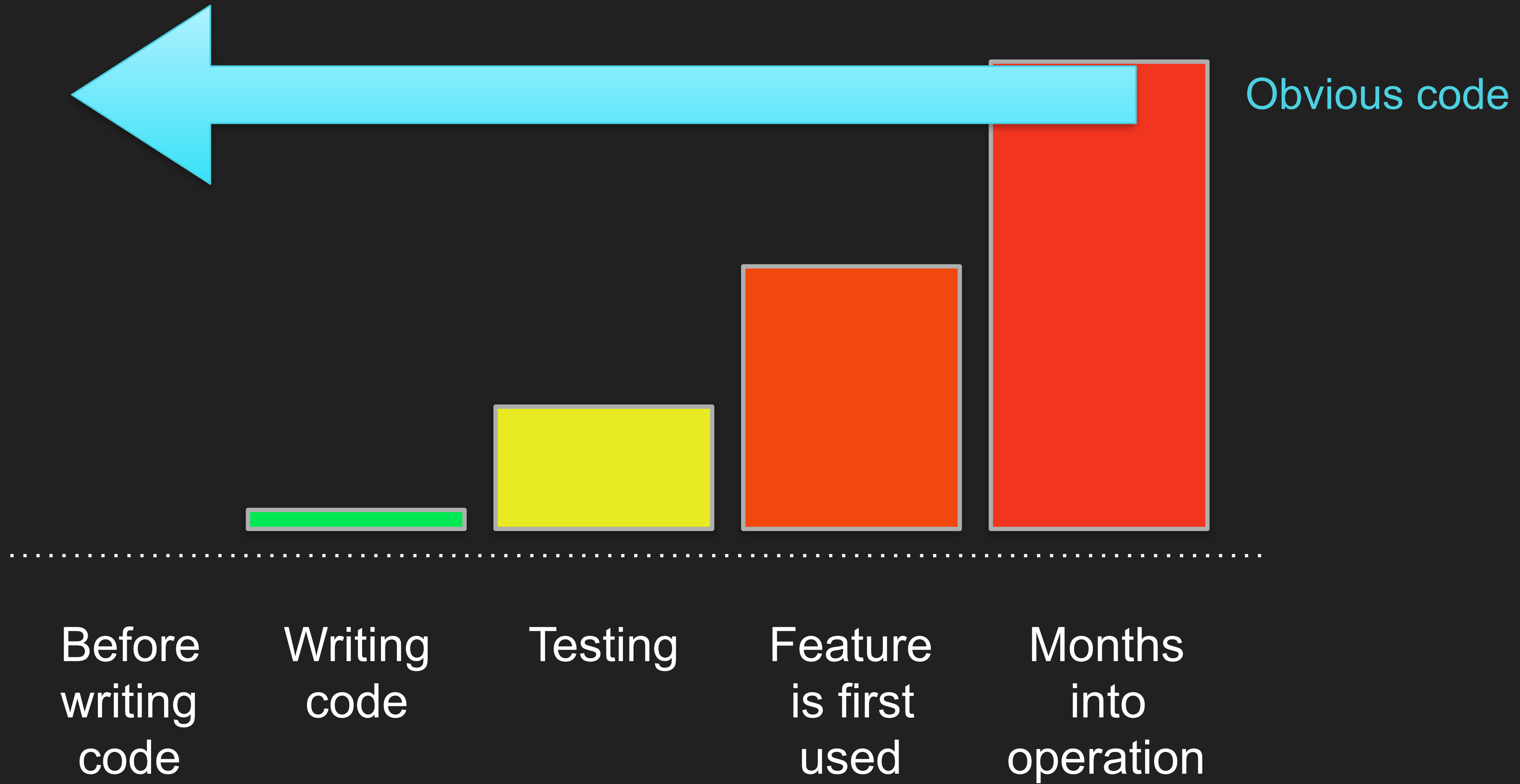
@daveliddament

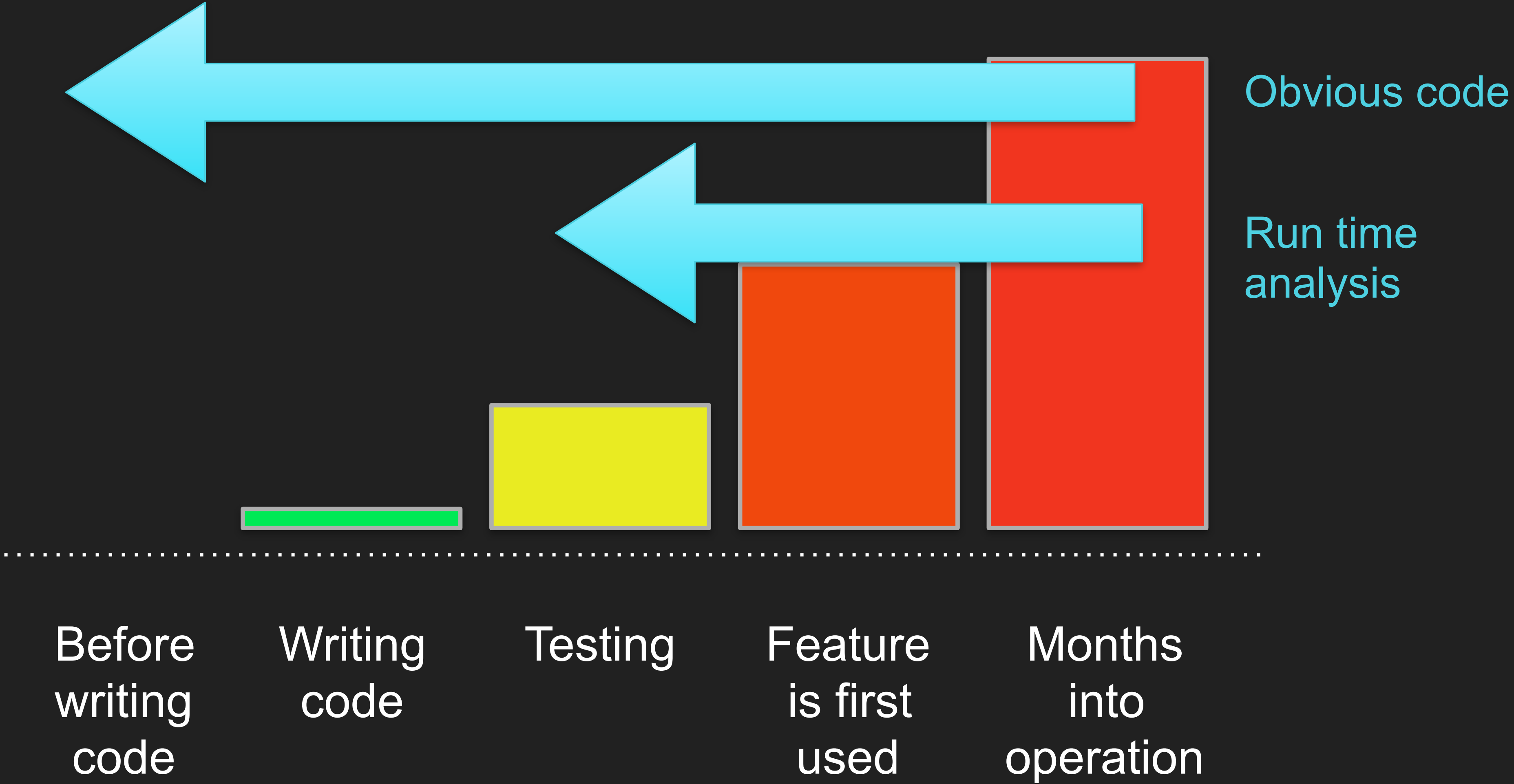
Lamp Bristol

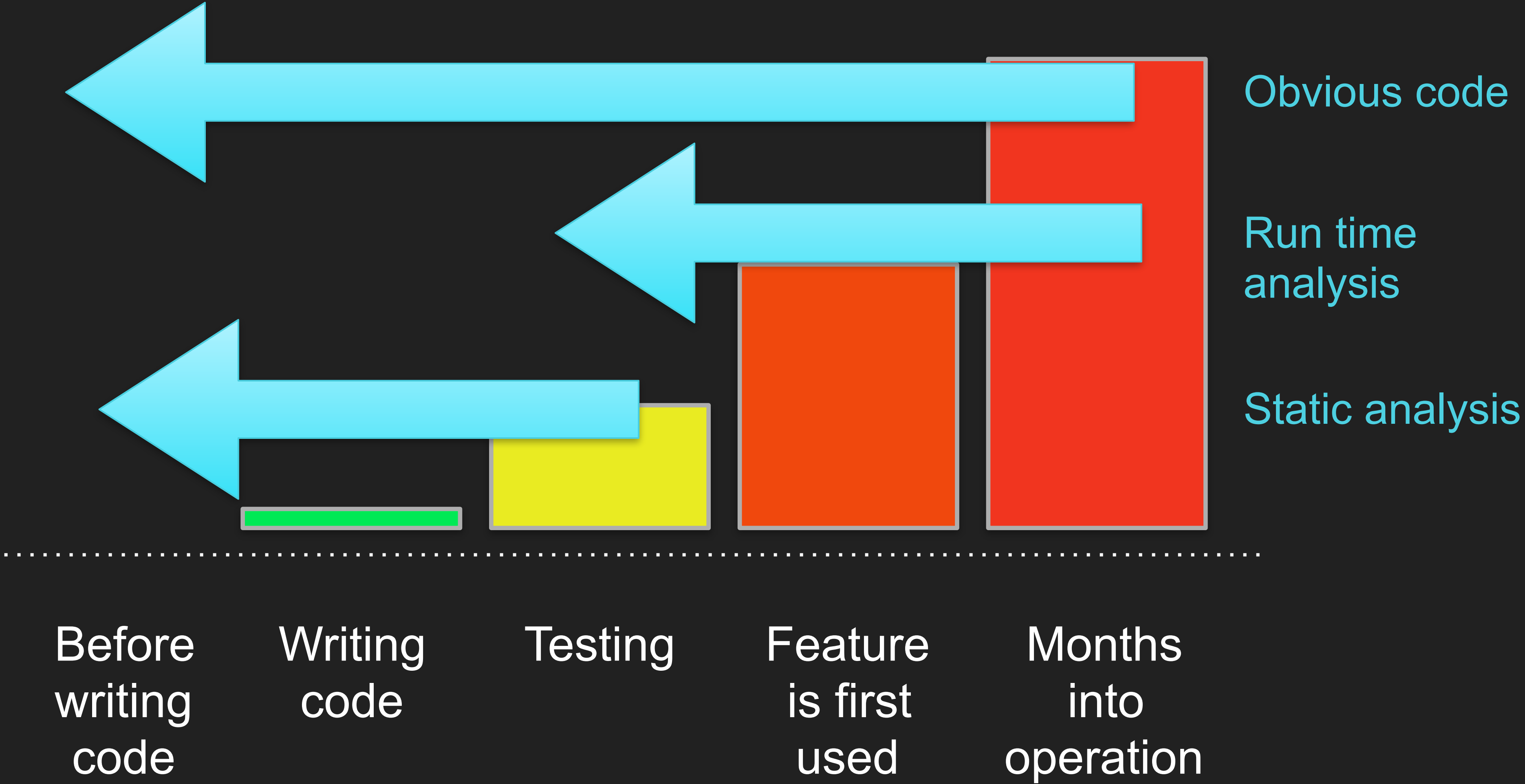
15+ years software development (PHP, Java, Python, C)

Organise PHP-SW user group and Bristol PHP Training


Before writing code
Writing code
Testing
Feature is first used
Months into operation


Obvious code
Before writing code
Writing code
Testing
Feature is first used
Months into operation


Obvious code
Run time analysis
Before writing code
Writing code
Testing
Feature is first used
Months into operation


Obvious code
Run time analysis
Static analysis
Before writing code
Writing code
Testing
Feature is first used
Months into operation

Static analysis

Is this code valid?

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

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

Is this code valid?

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

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

Is this code valid?

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

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

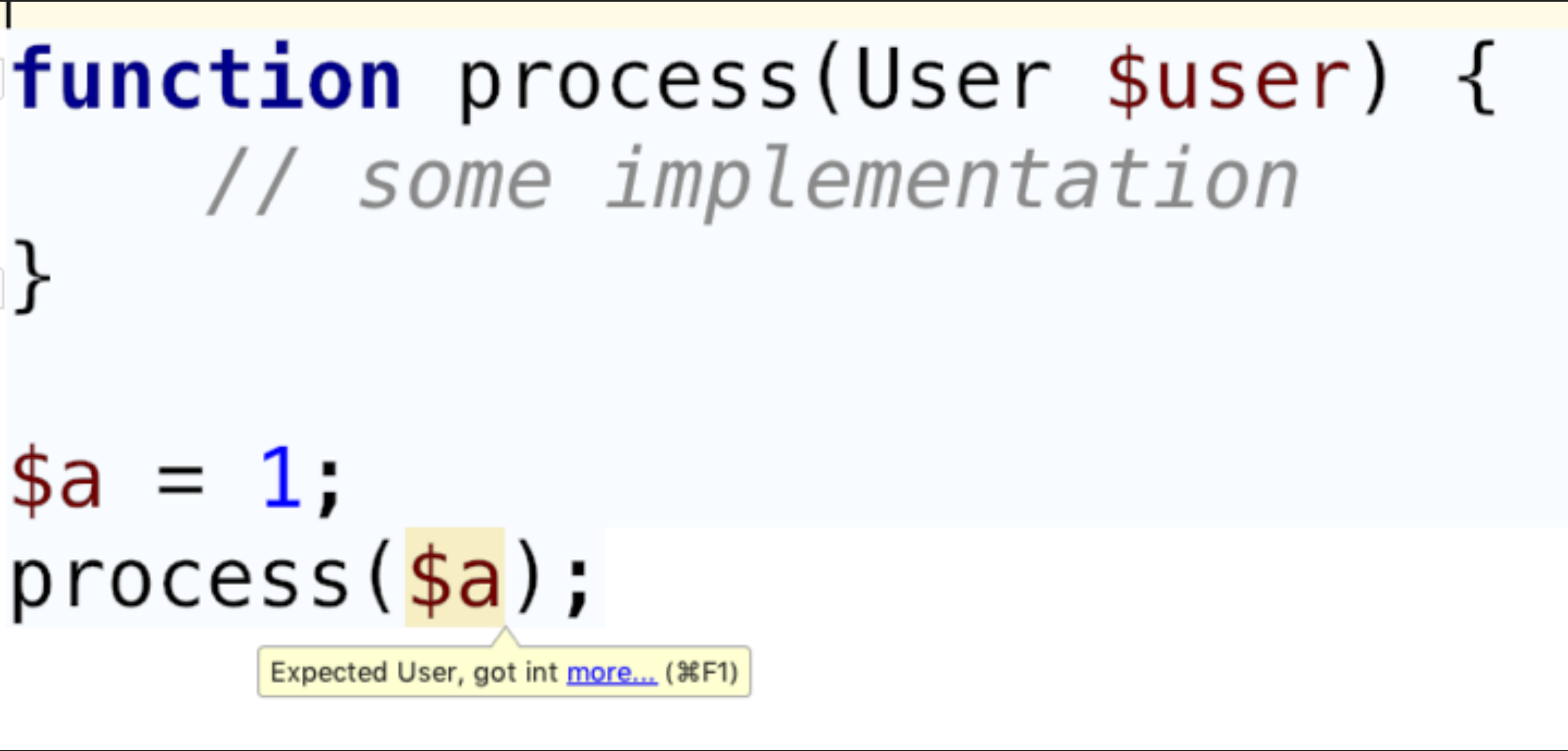
Is this code valid?

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

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

Is this code valid?

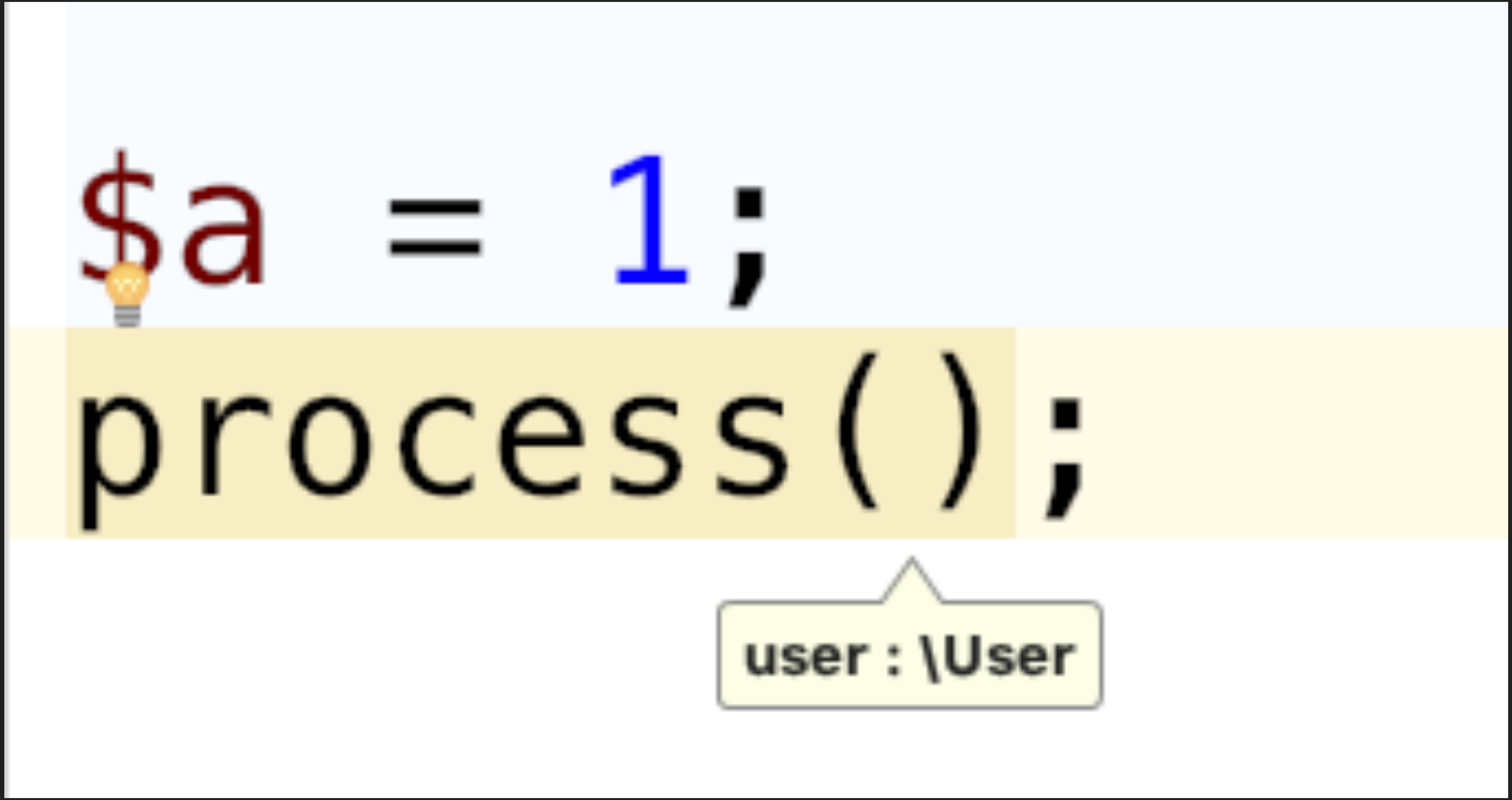
```
function process(User $user) {
 // some implementation
}

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

@DaveLiddament



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


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

Expected User, got int more... (⌘F1)

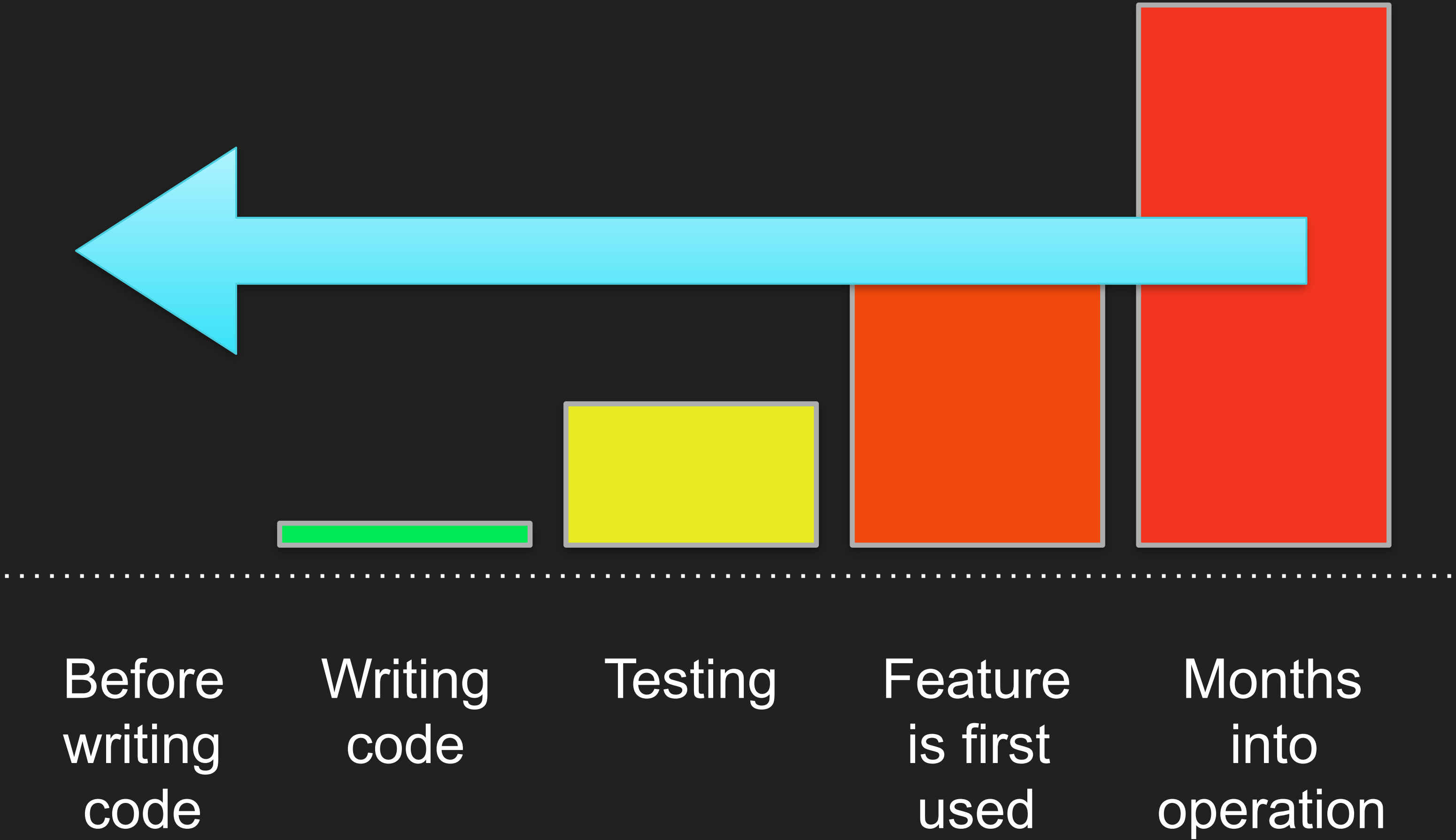


```
$a = 1;
process();
```

user : \User

Type hinting has helped

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

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


Before writing code
Writing code
Testing
Feature is first used
Months into operation

More type hinting with PHP 7

```
function duplicateString (
    string $value,
    int $times) :string
```

More type hinting with PHP 7

```
function duplicateString (
      string $value,
      int $times) :string
```

More type hinting with PHP 7

```
function duplicateString (
    string $value,
    int $times) :string
```

@DaveLiddament

Is this code valid?

```
function getUser(int $id): User {…}

function process(User $user): void {…}

$a = getUser(12);
process($a);
```

@DaveLiddament

Is this code valid?

```
function getUser(int $id): User {…}

function process(User $user): void {…}

$a = getUser(12);
process($a);
```

@DaveLiddament

Is this code valid?

```
function getUser(int $id): User {…}

function process(User $user): void {…}

$a = getUser(12);
process($a);
```

@DaveLiddament

Is this code valid?

```
function getUser(int $id): User {…}

function process(User $user): void {…}

$a = getUser(12);
process($a);
```

@DaveLiddament

Is this code valid?

```
function getUser(int $id): User {…}

function process(User $user): void {…}

$a = getUser(12);
process($a);
```

@DaveLiddament

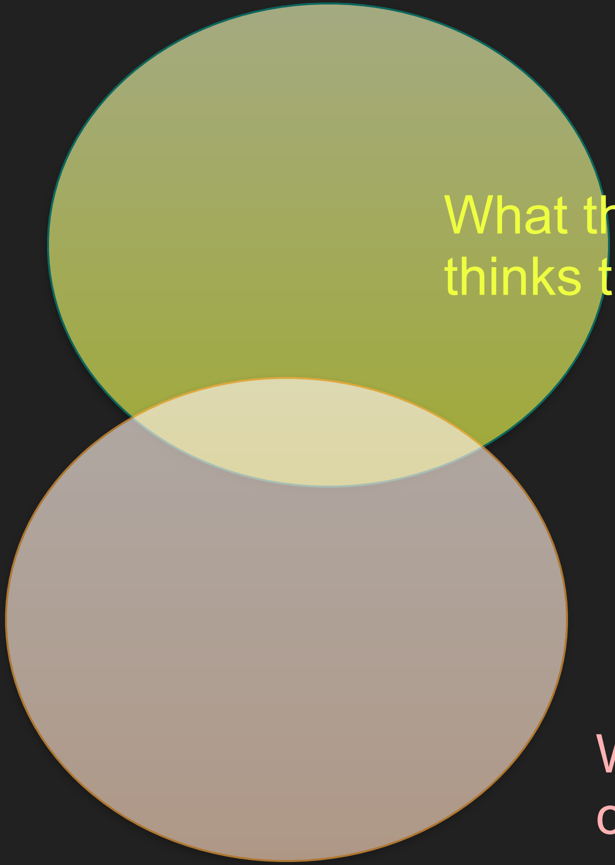
Is this code valid?

```
function getUser(int $id): User {…}

function process(User $user): void {…}

$a = getUser(12);
process($a);
```

Is this code valid?

```
function getUser(int $id): User {…}

function process(User $user): void {…}

$a = getUser(12);
process($a);
```

Win Win

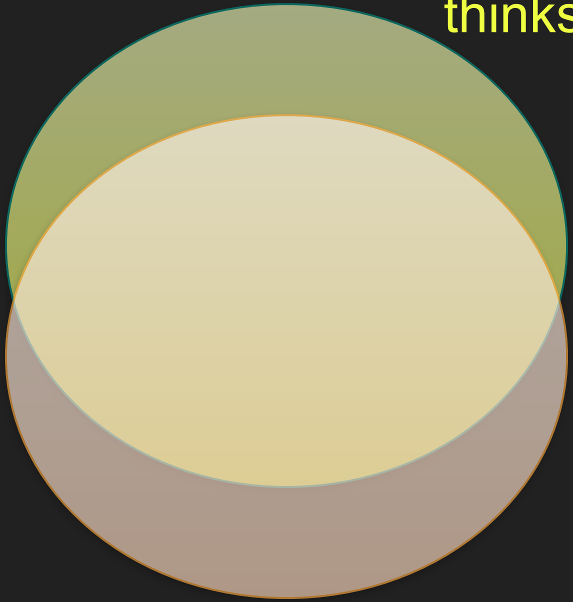


What the developer thinks the code does

What the code actually does

@DaveLiddament

Win Win


What the developer thinks the code does
What the code actually does

@DaveLiddament

Language level validation

```
function getUser(int $id): User {…}

function process(User $user): void {…}

$a = getUser(“dave”);
process($a);
```

Language level validation

```
function getUser(int $id): User {…}

function process(User $user): void {…}

$a = getUser(“dave”);
process($a);
```

Language level validation

```
function getUser(int $id): User {…}

function process(User $user): void {…}

$a = getUser(“dave”);
process($a);
```

Static analysis can cover language gaps

```
class User {
  public function getAccountNumber() :string {…}
}

/**
 * @return User[]
 */
function getUsers(): array { … }

$users = getUsers();
foreach($users as $user) {
  $accountNumber = $user->getAccountNumber();
}
```

Static analysis can cover language gaps

```
class User {
  public function getAccountNumber() :string {…}
}
```

```
/**
 * @return User[]
 */
function getUsers(): array { … }

$users = getUsers();
foreach($users as $user) {
  $accountNumber = $user->getAccountNumber();
}
```

Static analysis can cover language gaps

```
class User {
  public function getAccountNumber() :string {…}
}

/**
 * @return User[]
 */
function getUsers(): array { … }

$users = getUsers();
foreach($users as $user) {
  $accountNumber = $user->getAccountNumber();
}
```

@DaveLiddament

Static analysis can cover language gaps

```
class User {
  public function getAccountNumber() :string {…}
}

/**
 * @return User[]
 */
function getUsers(): array { … }

$users = getUsers();
foreach($users as $user) {
  $accountNumber = $user->getAccountNumber();
}
```

Static analysis can cover language gaps

```
class User {
  public function getAccountNumber() :string {…}
}

/**
 * @return User[]
 */
function getUsers(): array { … }

$users = getUsers();
foreach($users as $user) {
  $accountNumber = $user->getAccountNumber();
}
```

Static analysis can cover language gaps

```
class User {
  public function getAccountNumber() :string {…}
}

/**
 * @return User[]
 */
function getUsers(): array { … }

$users = getUsers();
foreach($users as $user) {
  $accountNumber = $user->getAccountNumber();
}
```

@DaveLiddament

Static analysis can cover language gaps

```
class User {
  public function getAccountNumber() :string {…}
}

/**
 * @return User[]
 */
function getUsers(): array { … }

$users = getUsers();
foreach($users as $user) {
  $accountNumber = $user->getAccountNumber();
}
```

Static analysis can find errors

```
class User {
  public function getAccountNumber() :string {…}
}

/**
 * @return User[]
 */
function getUsers(): array { … }

$users = getUsers();
foreach($users as $user) {
  $accountNumber = $user->getSomething();
}
```

@DaveLiddament

Static analysis can find errors

```
class User {
  public function getAccountNumber() :string {…}
}

/**
 * @return User[]
 */
function getUsers(): array { … }

$users = getUsers();
foreach($users as $user) {
  $accountNumber = $user->getSomething();
}
```

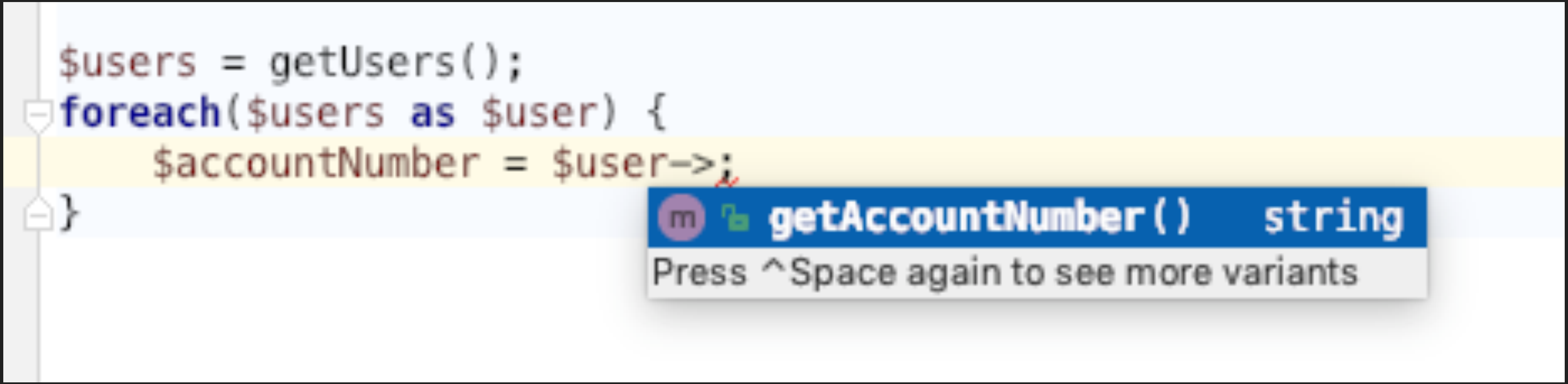
@DaveLiddament

Static analysis can find errors

```
class User {
  public function getAccountNumber() :string {…}
}

/**
 * @return User[]
 */
function getUsers(): array { … }

$users = getUsers();
foreach($users as $user) {
  $accountNumber = $user->getSomething();
}
```

@DaveLiddament

Static analysis helps developers



```
$users = getUsers();
foreach($users as $user) {
    $accountNumber = $user->;
}
```

m getAccountNumber() string
Press ^Space again to see more variants

Static analysis recap

Static analysis recap

- Analyse code without running it

Static analysis recap

- Analyse code without running it
- Prevent bugs even entering the code base

Static analysis recap

- Analyse code without running it
- Prevent bugs even entering the code base
- Type hinting and doc blocks comments help static analysis tools
    - which in turn help developers

@DaveLiddament

Static analysis recap

- Analyse code without running it
- Prevent bugs even entering the code base
- Type hinting and doc blocks comments help static analysis tools
  - which in turn help developers
- Use an IDE that offers static analysis

Run time checks

Run time checks

- Tests

Run time checks

- Tests
- Assertions

Tests are assertions

Tests are assertions

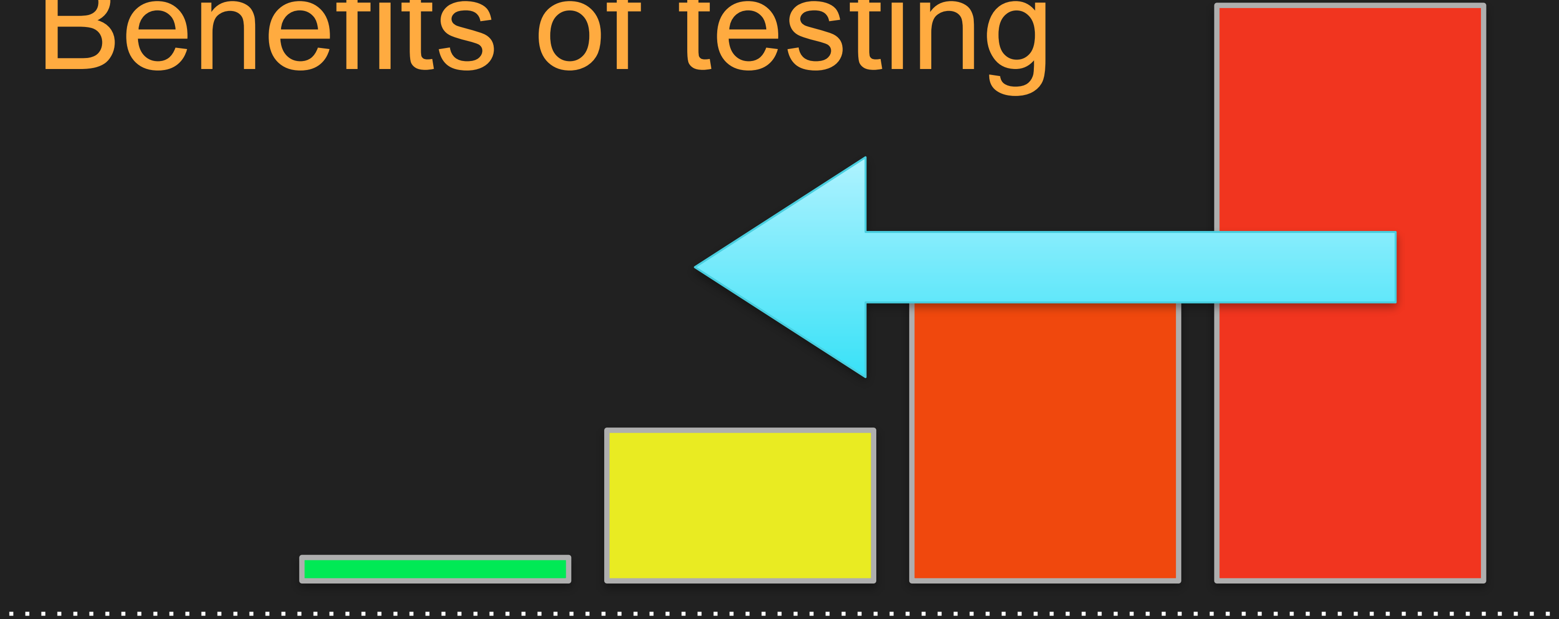
If I apply a discount code “SPEAKER”

Tests are assertions

If I apply a discount code “SPEAKER”

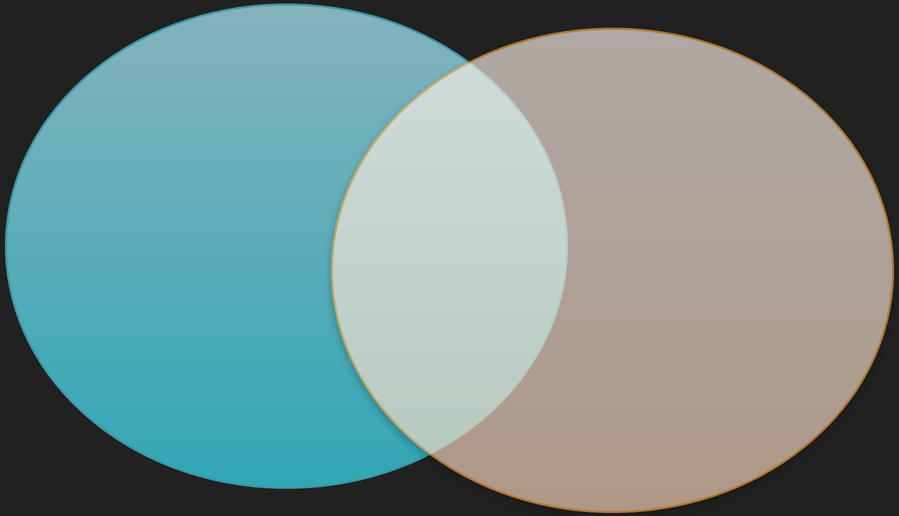
My conference ticket is reduced to £90

Benefits of testing



Before writing code	Writing code	Testing	Feature is first used	Months into operation

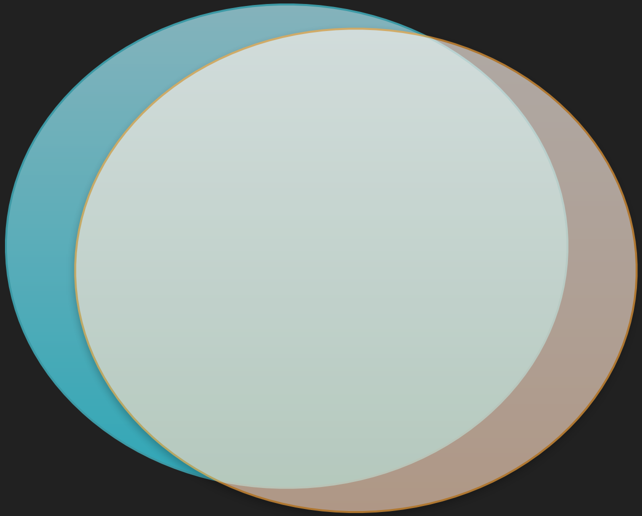
Benefits of testing



What the
code
should do

What the
code
actually does

Benefits of testing

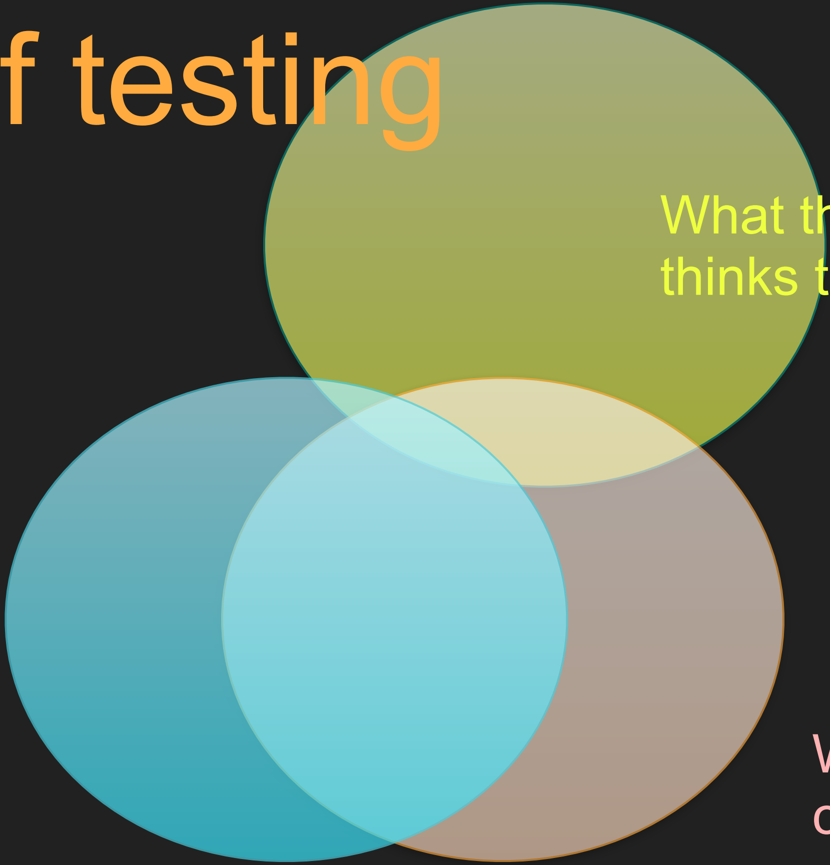


What the
code
should do

What the
code
actually does

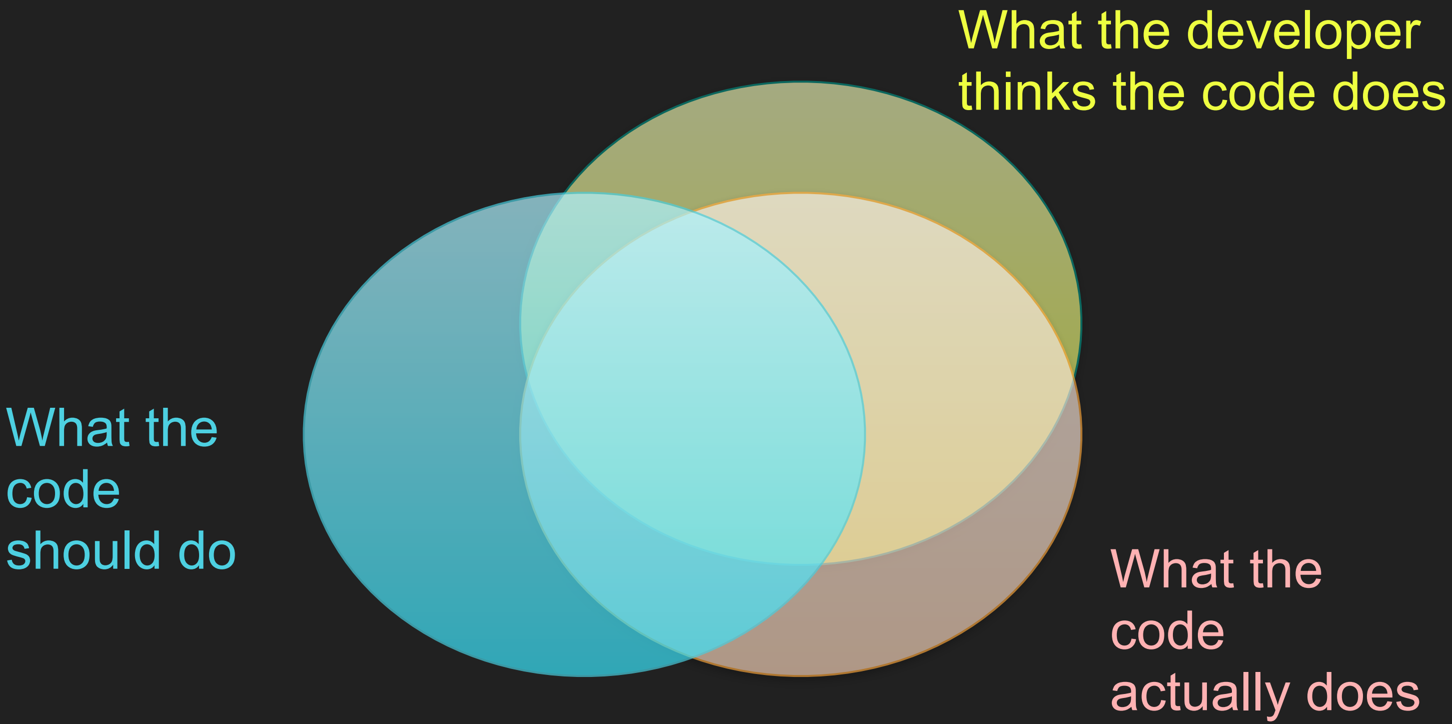
@DaveLiddament

Benefits of testing


What the developer thinks the code does
What the code should do
What the code actually does

@DaveLiddament

Benefits of testing


What the developer thinks the code does
What the code should do
What the code actually does

@DaveLiddament

Assertions in code

@DaveLiddament

Assertions in code

Statements that the developer believes should always be true

Code that worries me...

```
if ($type == 1) {
   $message = ‘hello’;
} elseif ($type == 2) {
   $message = ‘goodbye’;
}


sendMessage($message);
```

Code that worries me…

```
if ($type == 1) {
    $message = ‘hello’;
} elseif ($type == 2) {
    $message = ‘goodbye’;
}


sendMessage($message);
```

Code that worries me...

```
if ($type == 1) {
    $message = ‘hello’;
} elseif ($type == 2) {
    $message = ‘goodbye’;
}


sendMessage($message);
```

Code that worries me...

```
if ($type == 1) {
    $message = ‘hello’;
} elseif ($type == 2) {
    $message = ‘goodbye’;
}


sendMessage($message);
```

Now I’m happier…

```
if ($type == 1) {
    $message = ‘hello’;
} elseif ($type == 2) {
    $message = ‘goodbye’;
} else {
    throw new Exception(“Invalid type”);
}
sendMessage($message);
```

Now I’m happier…

```
if ($type == 1) {
    $message = ‘hello’;
} elseif ($type == 2) {
    $message = ‘goodbye’;
} else {
    throw new Exception(“Invalid type”);
}
sendMessage($message);
```

@DaveLiddament

Can we improve this code

```
public function setStatus(string $status){
    $this->status = $status;
}
```

Improvement 1: Add constants

```
const REGISTERED = ‘registered’;
const STARTED = ‘started’;
const FINISHED = ‘finished’;
const QUIT = ‘quit’;

public function setStatus(string $status){
    $this->status = $status;
}
```

Improvement 1: Add constants

```
const REGISTERED = ‘registered’;
const STARTED = ‘started’;
const FINISHED = ‘finished’;
const QUIT = ‘quit’;

public function setStatus(string $status){
    $this->status = $status;
}
```

Improvement 2: Add assertion

```
… constants defined as before …

public function setStatus(string $status){
  if (!in_array($status,[self::REGISTERED,
        self::STARTED, self::FINISHED]) {
    throw new Exception(“Invalid status”);
  }
  $this->status = $status;
}
```

Improvement 2: Add assertion

```
… constants defined as before …

public function setStatus(string $status){
  if (!in_array($status,[self::REGISTERED,
        self::STARTED, self::FINISHED]) {
    throw new Exception(“Invalid status”);
  }
  $this->status = $status;
}
```

Improvement 2: Add assertion

```
… constants defined as before …

public function setStatus(string $status){
  if (!in_array($status,[self::REGISTERED,
          self::STARTED, self::FINISHED]) {
    throw new Exception(“Invalid status”);
  }
  $this->status = $status;
}
```

@DaveLiddament

Improving error messages

Improving error messages

Invalid type

Improving error messages

Invalid type

Invalid type [$type]

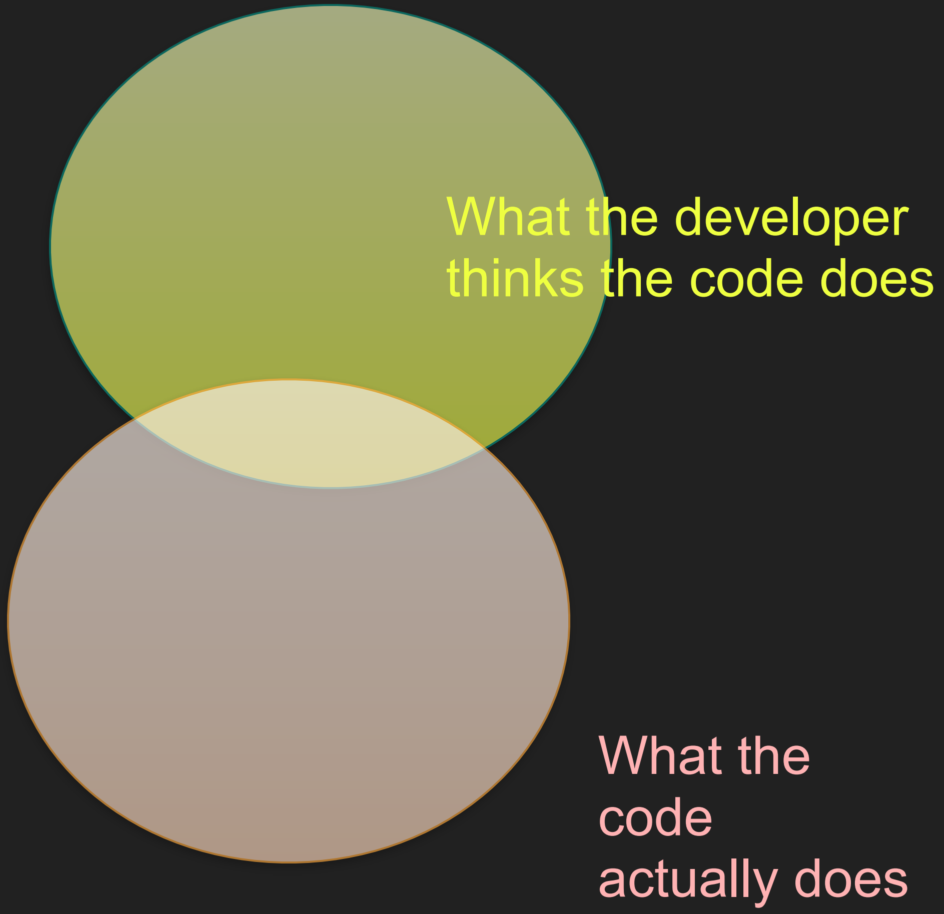
Improving error messages

Invalid type

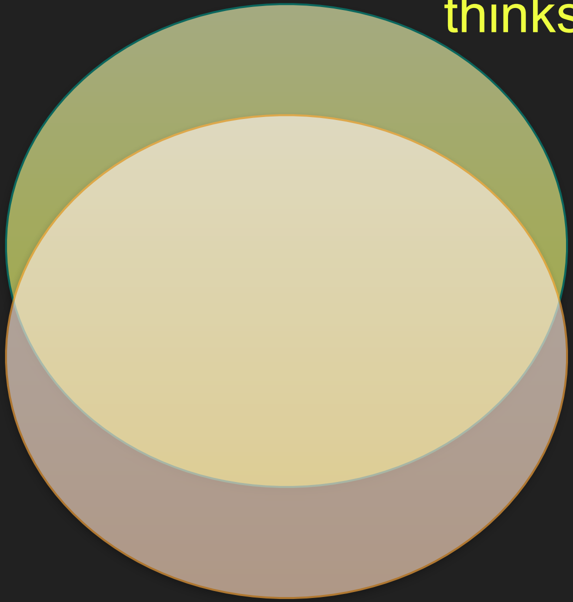
Invalid type [$type]

Invalid type [$type] for user [$userId]

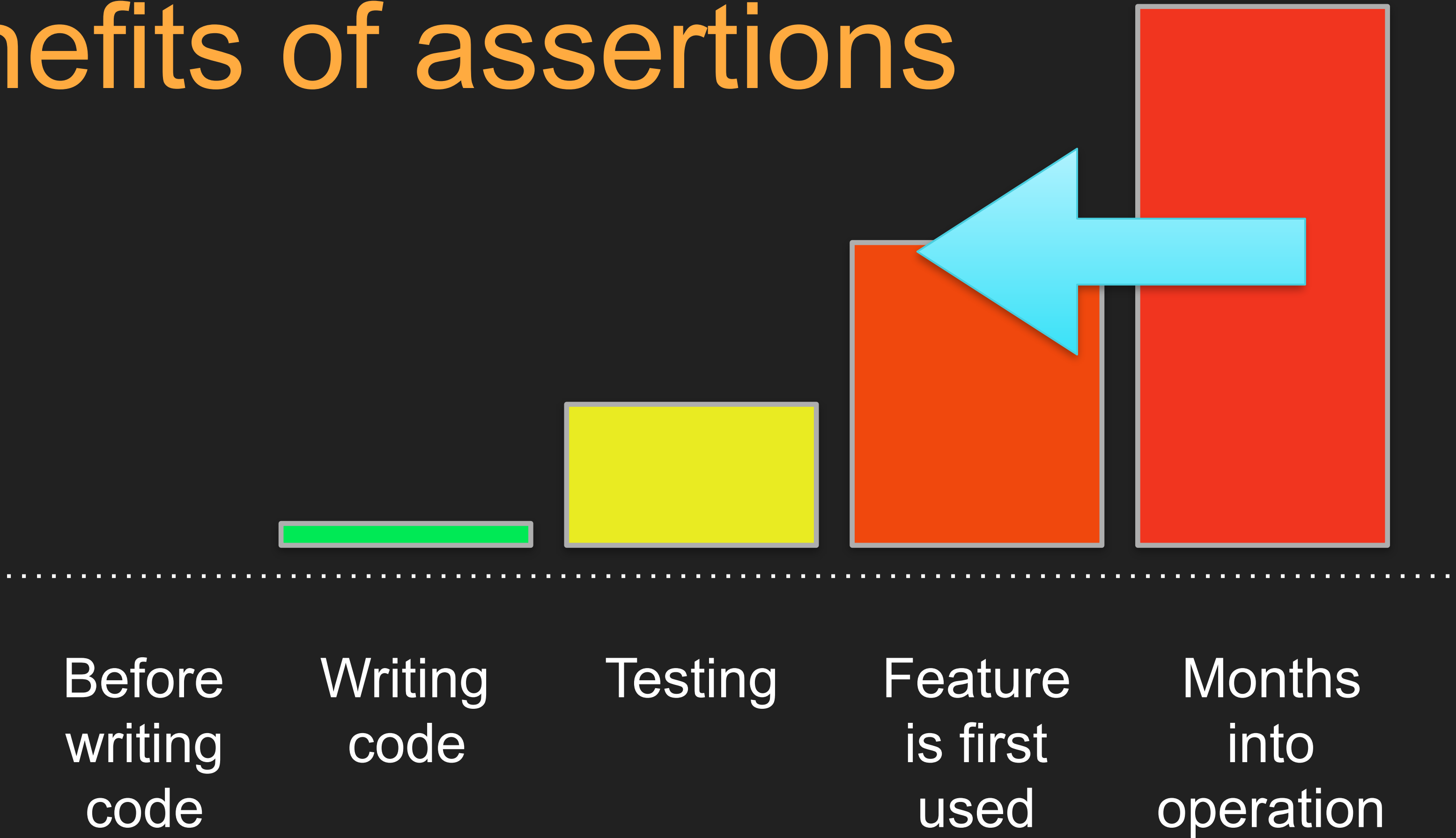
Wins from asserts


What the developer thinks the code does
What the code actually does

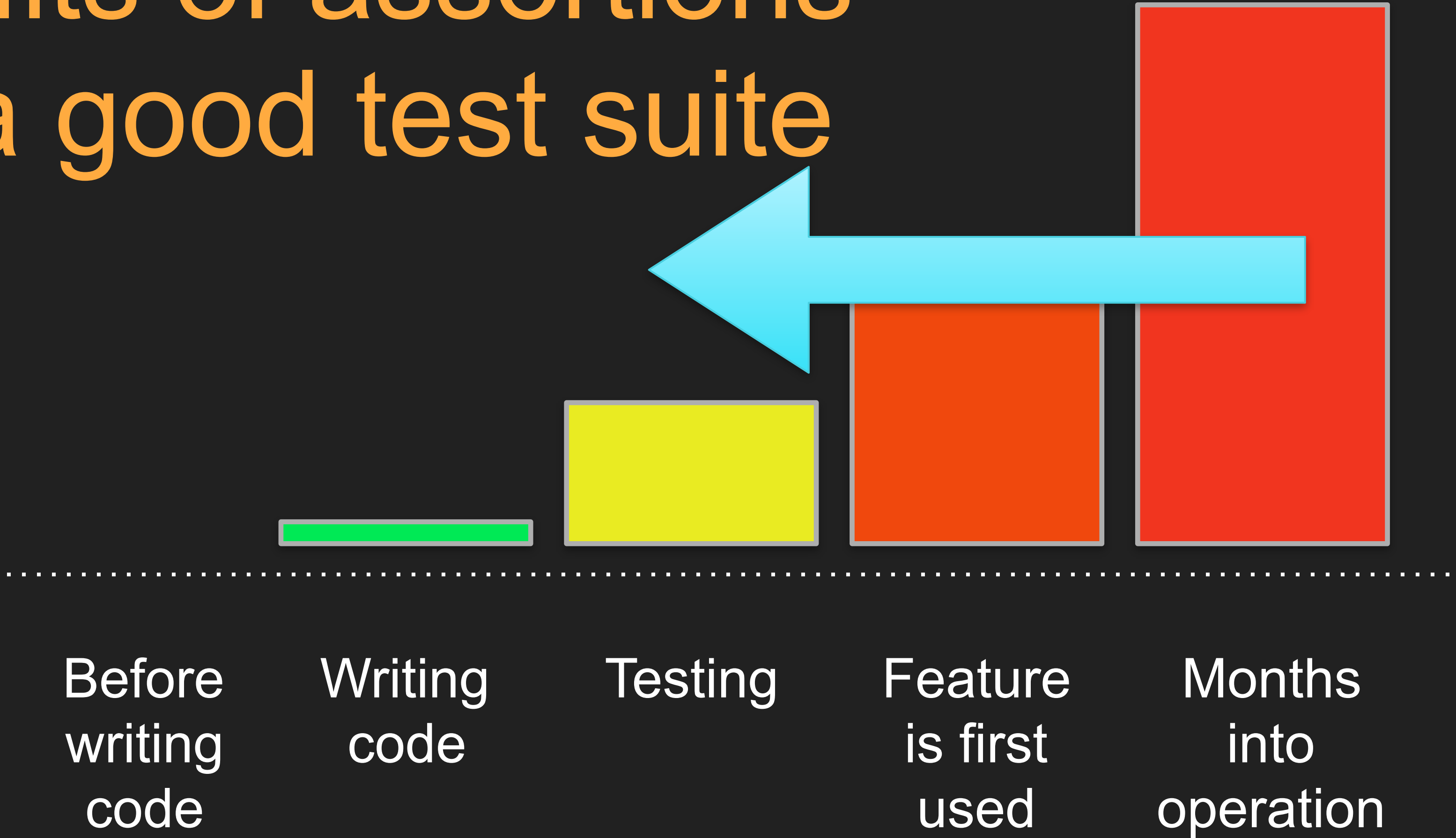
Wins from asserts


What the developer thinks the code does
What the code actually does

Benefits of assertions



Benefits of assertions and a good test suite



Obvious code

Obvious code

- Value Objects

Obvious code

- Value Objects
- Rename and Refactor

Value Objects

Can we improve this code?

```
class MarketingCampaign {

  … some methods …

  public function addAddress(string $address);
}


$campaign = new MarketingCampaign();
$campaign->addAddress(“dave@phpsw.uk”)
```

Can we improve this code?

```
class MarketingCampaign {

  … some methods …

  public function addAddress(string $address);
}


$campaign = new MarketingCampaign();
$campaign->addAddress(“dave@phpsw.uk”)
```

These are all strings…

dave@phpsw.uk

fredblogs.com

fred.blogs

fred@blogs.com

6 Lower Park Row, Bristol

These are all strings…

dave@phpsw.uk

fredblogs.com

fred.blogs

fred@blogs.com

6 Lower Park Row, Bristol

This is wrong (and our IDE can’t spot mistake)

```
class MarketingCampaign {

  .. some methods ..

  public function addAddress(string $address);
}


$campaign = new MarketingCampaign();
$campaign->addAddress(“6 Lower Park Row, Bristol”)
```

This is wrong (and our IDE can't spot mistake)

```
class MarketingCampaign {

  .. some methods ..

  public function addAddress(string $address);
}


$campaign = new MarketingCampaign();
$campaign->addAddress("6 Lower Park Row, Bristol")
```

@DaveLiddament

EmailAddress object instead of primitive

```
class EmailAddress {

  private $emailAddress;

  public function __construct(string $emailAddress) {
    $this->emailAddress = $emailAddress;
  }

  public function getEmailAddress(): string {
    return $this->emailAddress;
  }
}
```

EmailAddress object instead of primitive

```
class EmailAddress {

  private $emailAddress;

  public function __construct(string $emailAddress) {
    $this->emailAddress = $emailAddress;
  }

  public function getEmailAddress(): string {
    return $this->emailAddress;
  }
}
```

EmailAddress object instead of primitive

```
class EmailAddress {

  private $emailAddress;

  public function __construct(string $emailAddress) {
    $this->emailAddress = $emailAddress;
  }

  public function getEmailAddress(): string {
    return $this->emailAddress;
  }
}
```

EmailAddress object instead of primitive

```
class EmailAddress {

  private $emailAddress;

  public function __construct(string $emailAddress) {
    $this->emailAddress = $emailAddress;
  }

  public function getEmailAddress(): string {
    return $this->emailAddress;
  }
}
```

Using EmailAddress

```
class MarketingCampaign {

  .. some methods ..

  public function addAddress(EmailAddress $address);
}


$campaign = new MarketingCampaign();
$emailAddress = new EmailAddress(“dave@phpsw.uk”)
$campaign->addAddress($emailAddress)
```

Using EmailAddress

```
class MarketingCampaign {

  .. some methods ..

  public function addAddress(EmailAddress $address);
}


$campaign = new MarketingCampaign();
$emailAddress = new EmailAddress(“dave@phpsw.uk”)
$campaign->addAddress($emailAddress)
```

Using EmailAddress

```
class MarketingCampaign {

   .. some methods ..

   public function addAddress(EmailAddress $address);
}


$campaign = new MarketingCampaign();
$emailAddress = new EmailAddress(“dave@phpsw.uk”)
$campaign->addAddress($emailAddress)
```

Using EmailAddress

```
class MarketingCampaign {

  .. some methods ..

  public function addAddress(EmailAddress $address);
}


$campaign = new MarketingCampaign();
$emailAddress = new EmailAddress(“dave@phpsw.uk”)
$campaign->addAddress($emailAddress)
```

This will fail (and your IDE will warn you)

```
class MarketingCampaign {

  .. some methods ..

  public function addAddress(EmailAddress $address);
}


$campaign = new MarketingCampaign();
$campaign->addAddress(“6 Lower Park Row, Bristol”)
```

This will fail (and your IDE will warn you)

```
class MarketingCampaign {

  .. some methods ..

  public function addAddress(EmailAddress $address);
}


$campaign = new MarketingCampaign();
$campaign->addAddress("6 Lower Park Row, Bristol")
```

But this is wrong

```
$emailAddress = new EmailAddress(“6 Lower Park Row”);
```

But this is wrong

```
$emailAddress = new EmailAddress(“6 Lower Park Row”);
```

@DaveLiddament

Add validation

```
public function __construct(string $emailAddress) {

  if ( … check email address is valid… == false) {
    throw new Exception(
      “Invalid email address [$emailAddress]”);
  }

  $this->emailAddress = $emailAddress;
}
```

Add validation

```
public function __construct(string $emailAddress) {

  if ( … check email address is valid… == false) {
    throw new Exception(
      “Invalid email address [$emailAddress]”);
  }

  $this->emailAddress = $emailAddress;
}
```

@DaveLiddament

Big win

We're guaranteed that EmailAddress represents a correctly formatted email address.

Are these email addresses the same?

dave@phpsw.uk

DAVE@phpsw.uk

DAVE@phpsw.UK

dave@PHPSW.uk

Store canonical form

```
public function __construct(string $emailAddress) {

  … validate email address …

  $this->emailAddress = $this->getCanonical($emailAddress);
}
```

Store canonical form

```
public function __construct(string $emailAddress) {
  … validate email address …
  $this->emailAddress = $this->getCanonical($emailAddress);
}
```

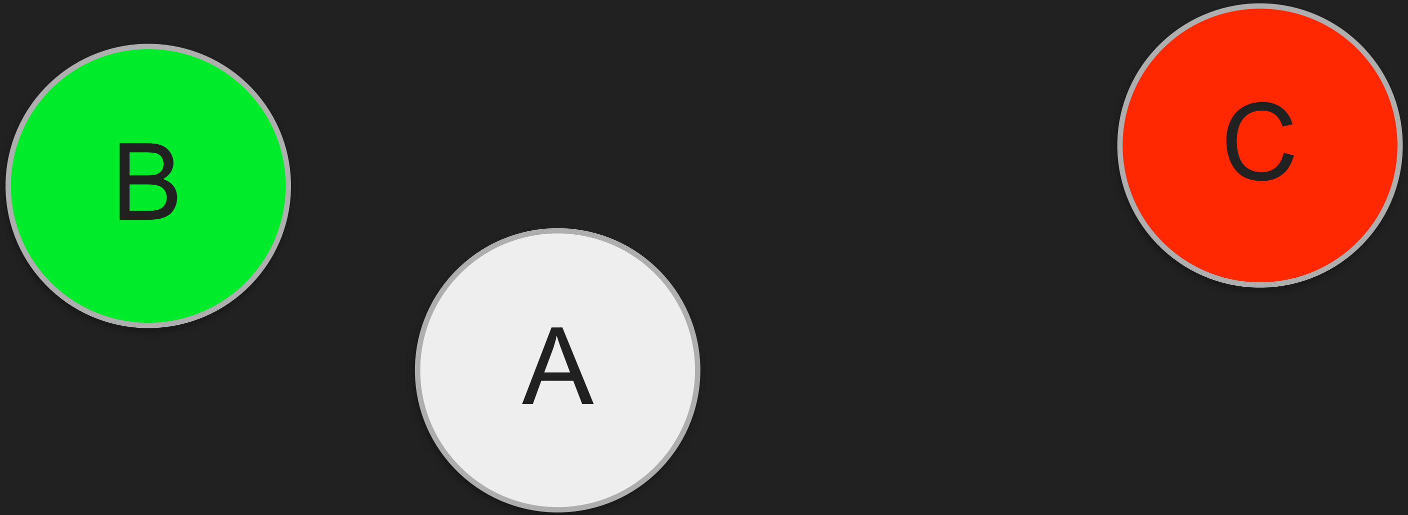
Postcodes formats

Normal:	`B1 1AB`
No spaces:	`B11AB`
Fixed width:	`B1   1AB`

Add domain specific logic

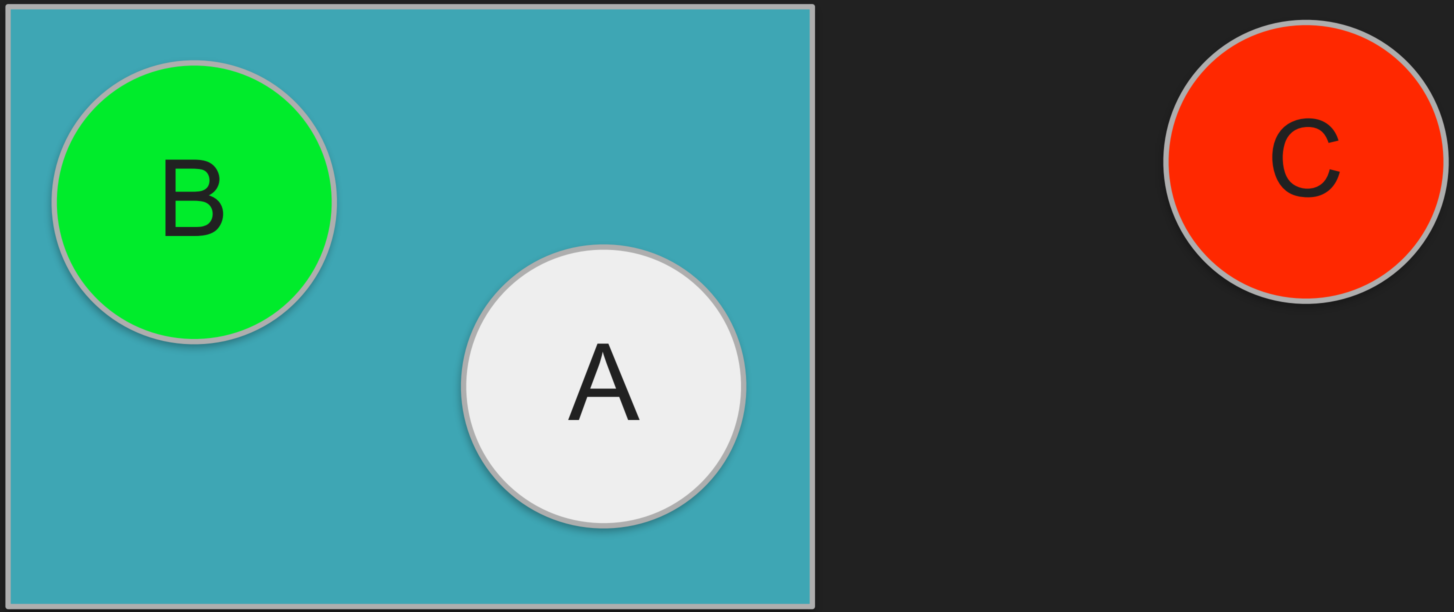
```
public function getPostcode(): string  {…}

public function getNoSpacesPostcode(): string {…}

public function getFixedWidthPostcode(): string {…}
```

Are these positions equal?



@DaveLiddament

Are these positions equal?



Add equals method

```
class Point
{
  const TOLERANCE = 10;

  … Other methods …

  public function equals(Point $other): bool
  {
     $distance = calculateDistance($this, $other);
     return $distance < self::TOLERANCE;
  }
}
```

Add equals method

```
class Point
{
  const TOLERANCE = 10;

  … Other methods …

  public function equals(Point $other): bool
  {
     $distance = calculateDistance($this, $other);
     return $distance < self::TOLERANCE;
  }
}
```

Be careful comparing objects…

```
if ($point1 == $point2) {
   … some code …
}


if ($point1->equals($point2)) {
   … some code …
}
```

@DaveLiddament

Be careful comparing objects…

```
if ($point1 == $point2) {
    … some code …
}
```

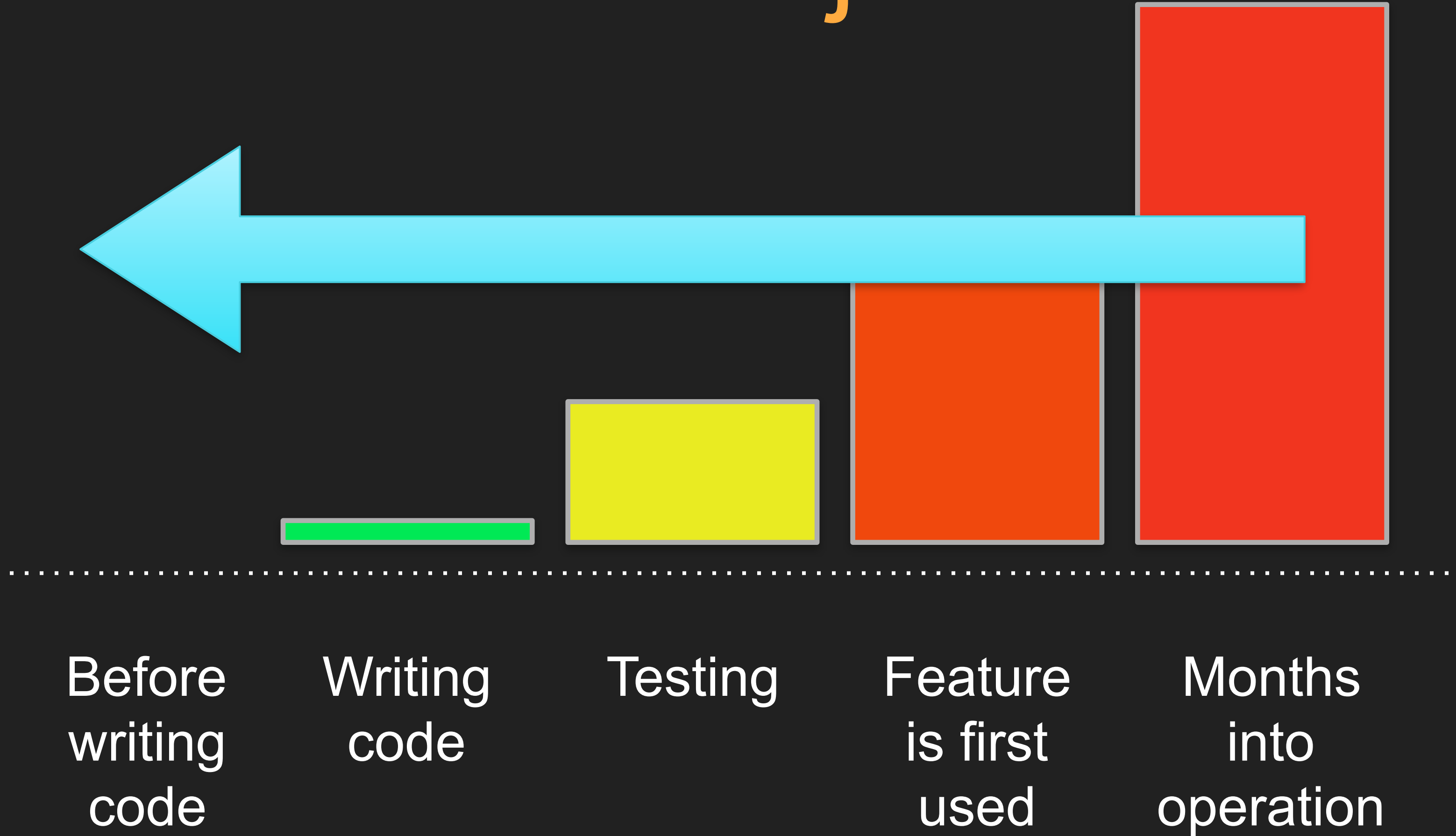
```
if ($point1->equals($point2)) {
    … some code …
}
```

Be careful comparing objects…

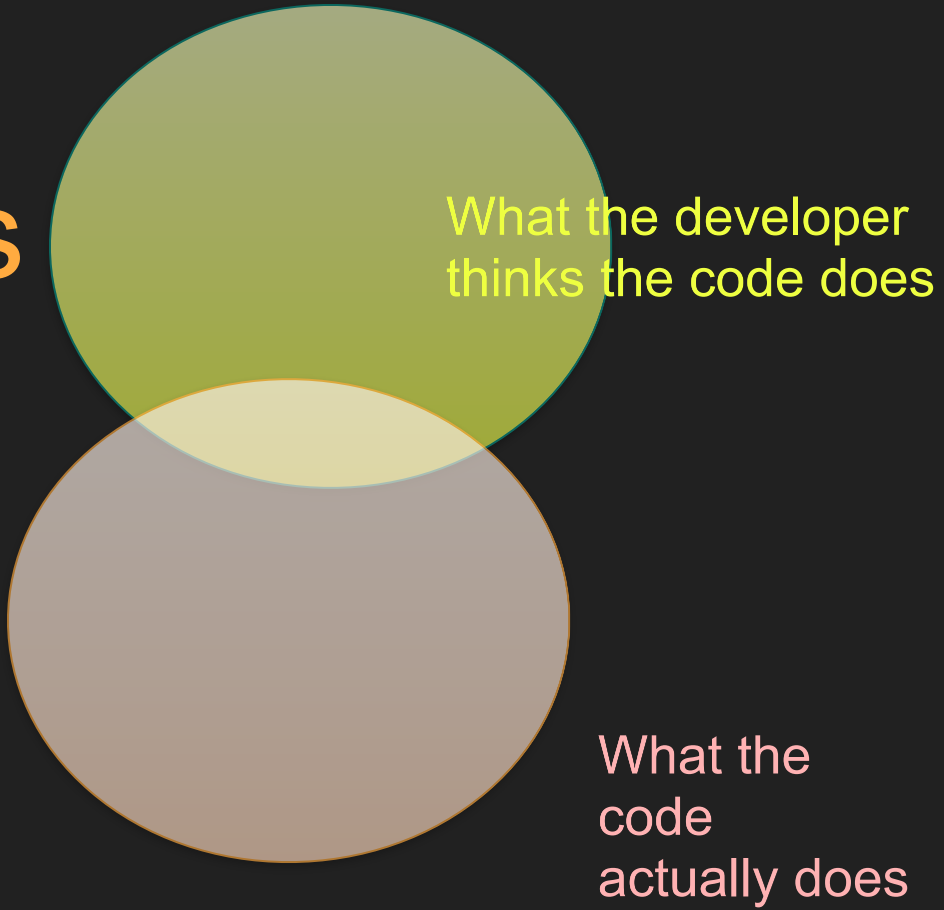
```
if ($point1 == $point2) {
    … some code …
}
```

```
if ($point1->equals($point2)) {
    … some code …
}
```

Benefits of Value Objects

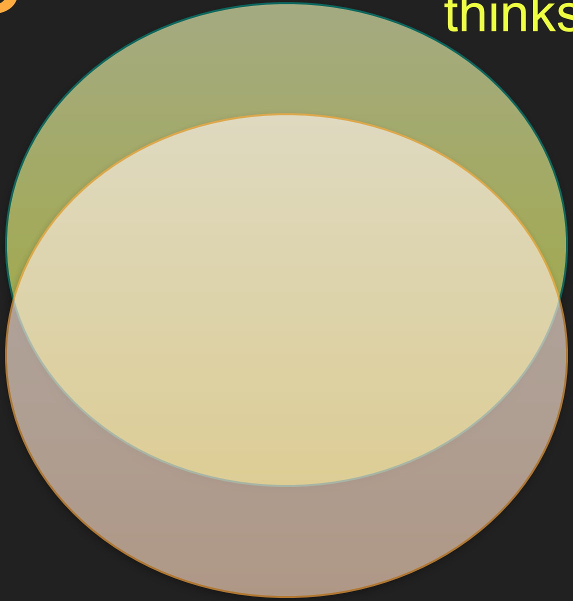


Wins from Value Objects


What the developer thinks the code does
What the code actually does

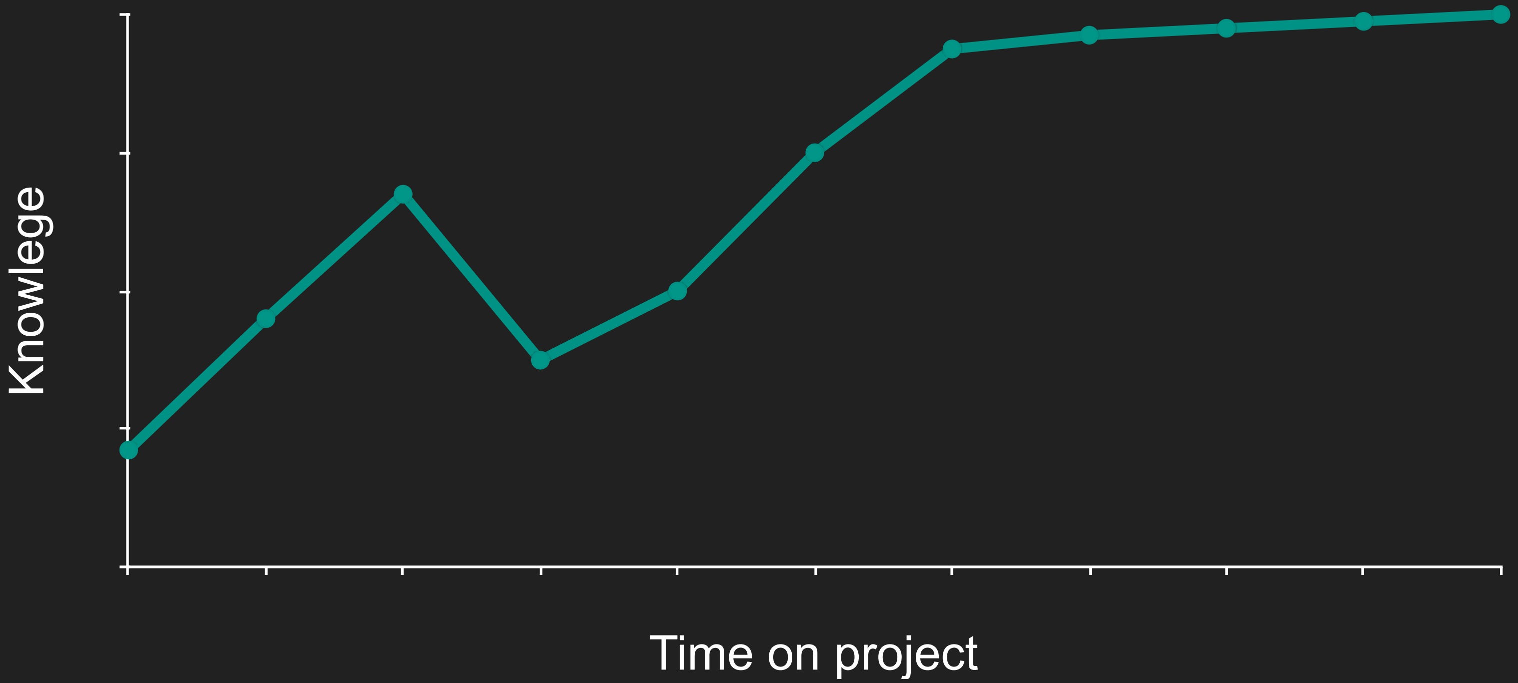
@DaveLiddament

Wins from Value Objects


What the developer thinks the code does
What the code actually does

@DaveLiddament

Rename and Refactor


Knowledge
Time on project

Renaming

```
class User {
 public function getName() {…}
}


class Game {
 public function getName() {…}
}
```

Renaming

```
class User {
 public function getName() {…}
}


class Game {
 public function getName() {…}
}
```

Renaming

```
class User {
 public function getName() {…}
}


class Game {
 public function getQuest() {…}
}
```

Renaming

```
class User {
 public function getName() {…}
}


class Game {
 public function getQuest() {…}
}
```

Renaming

```
function getUser();

function getGame();


$user = getUser();
$game = getGame();

echo ‘Hello ‘ . $user->getName();
echo ‘You are playing ‘ . $game->getName();
```

Renaming

```
function getUser();

function getGame();


$user = getUser();
$game = getGame();

echo ‘Hello ‘ . $user->getName();
echo ‘You are playing ‘ . $game->getName();
```

@DaveLiddament

Renaming

```
function getUser(): User;

function getGame(): Game;


$user = getUser();
$game = getGame();

echo ‘Hello ‘ . $user->getName();
echo ‘You are playing ‘ . $game->getQuest();
```

Renaming

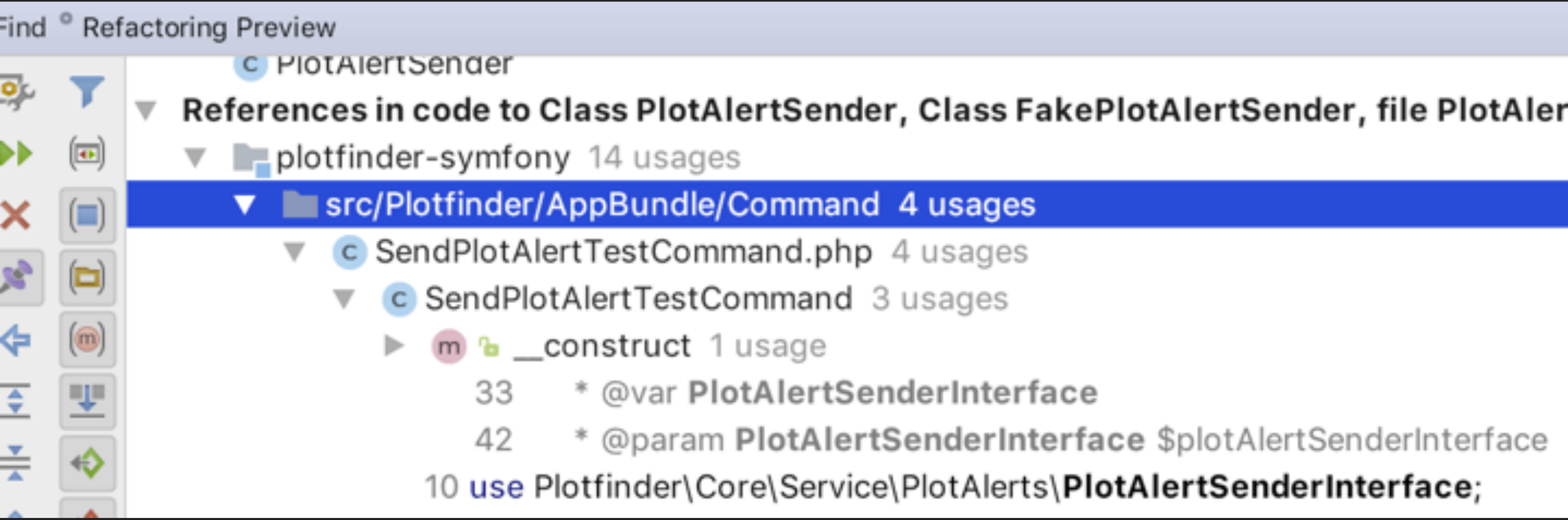
```
function getUser(): User;

function getGame(): Game;


$user = getUser();
$game = getGame();

echo ‘Hello ‘ . $user->getName();
echo ‘You are playing ‘ . $game->getQuest();
```

Renaming

```
function getUser(): User;

function getGame(): Game;


$user = getUser();
$game = getGame();

echo ‘Hello ‘ . $user->getName();
echo ‘You are playing ‘ . $game->getQuest();
```

@DaveLiddament

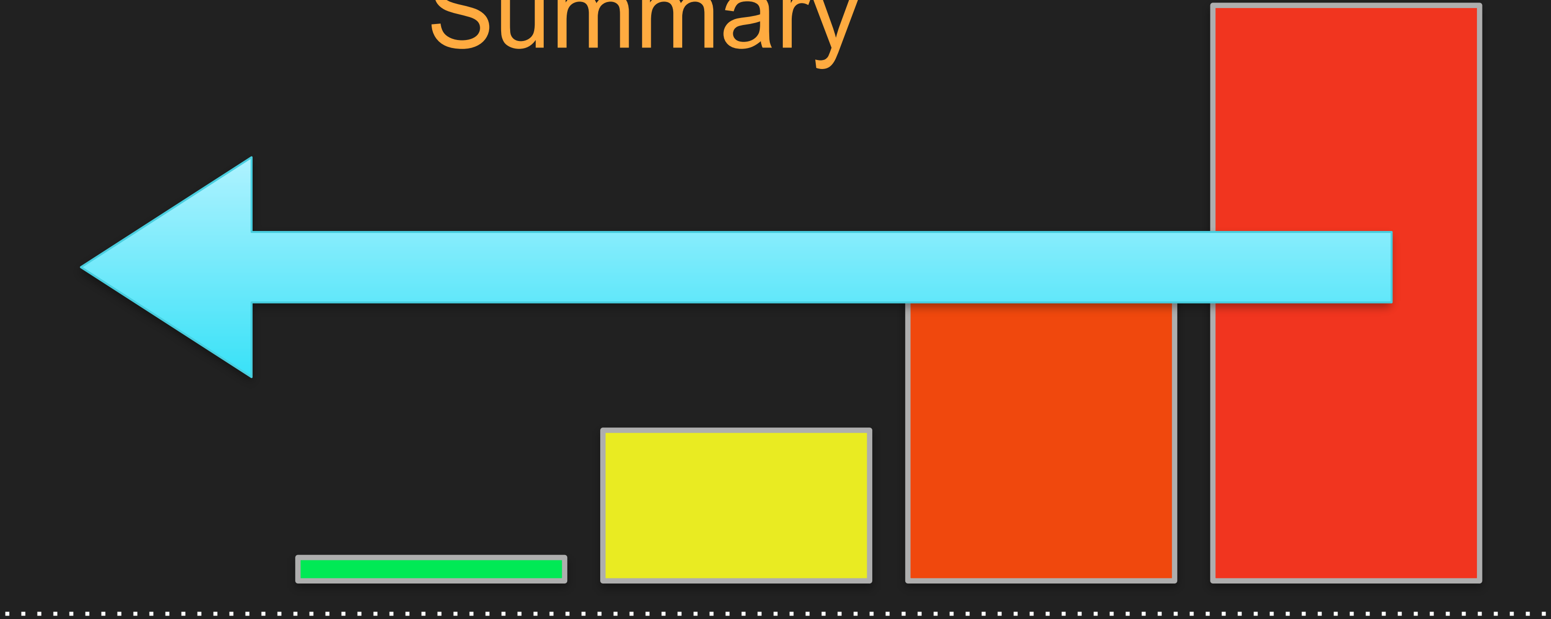
Win-Win: Rename and refactor


Find Refactoring Preview

```
PlotAlertSender
References in code to Class PlotAlertSender, Class FakePlotAlertSender, file PlotAler
  plotfinder-symfony 14 usages
    src/Plotfinder/AppBundle/Command 4 usages
      SendPlotAlertTestCommand.php 4 usages
        SendPlotAlertTestCommand 3 usages
          __construct 1 usage
            33   * @var PlotAlertSenderInterface
            42   * @param PlotAlertSenderInterface $plotAlertSenderInterface
        10 use Plotfinder\Core\Service\PlotAlerts\PlotAlertSenderInterface;
```

@DaveLiddament

Summary



Before writing code

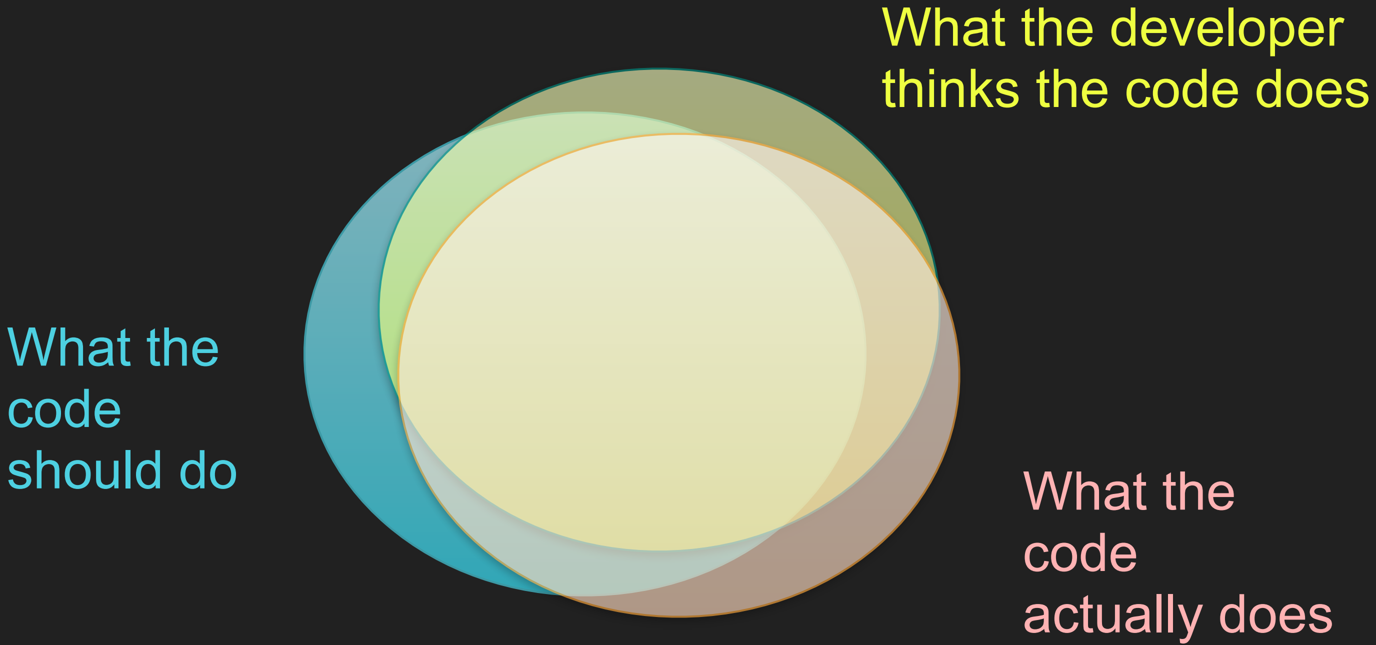
Writing code

Testing

Feature is first used

Months into operation

Summary


What the developer thinks the code does
What the code should do
What the code actually does

Summary

Summary

- Type hint everything you can

Summary

- Type hint everything you can
- Use docblock for language gaps

Summary

- Type hint everything you can
- Use docblock for language gaps
- Write tests

Summary

- Type hint everything you can
- Use docblock for language gaps
- Write tests
- Add assertions

Summary

- Type hint everything you can
- Use docblock for language gaps
- Write tests
- Add assertions
- Use Value Objects

Summary

- Type hint everything you can
- Use docblock for language gaps
- Write tests
- Add assertions
- Use Value Objects
- Rename and refactor

Summary

- Use a modern IDE

@DaveLiddament

Questions

Advice for improving

https://joind.in/talk/8de76

@DaveLiddament