Don’t be so primitive

Dave Liddament

@DaveLiddament

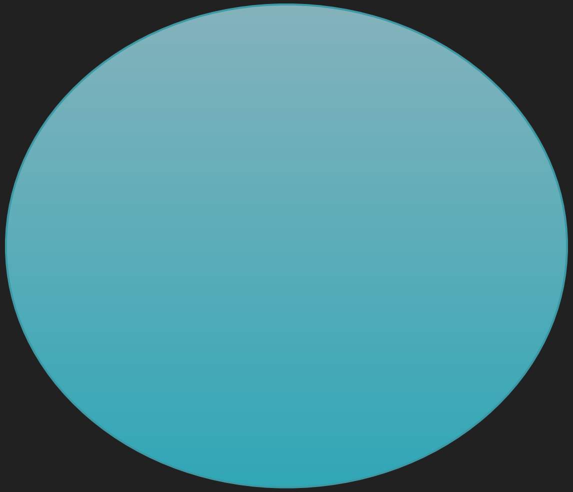
First let’s talk about bugs….

Question 1:
Who puts bugs in their code?

Why do bugs happen?

Why do bugs happen?

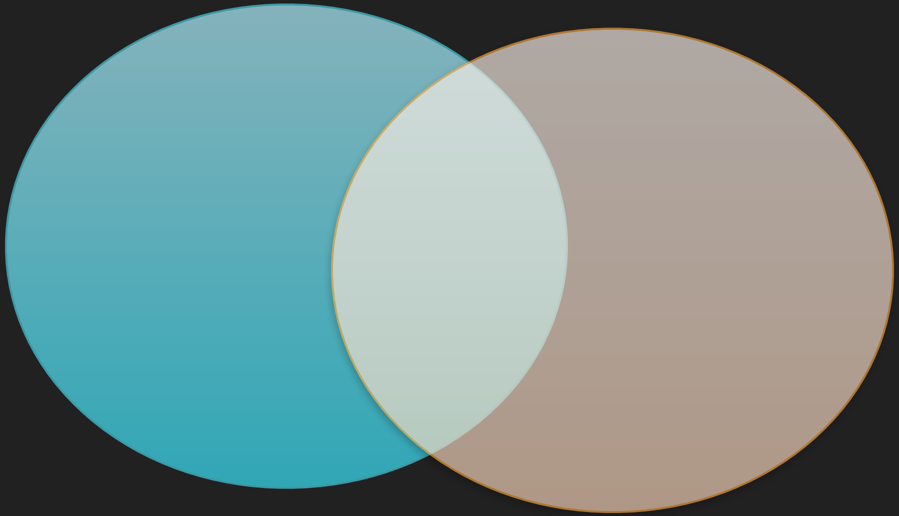
What the code should do



@DaveLiddament

Why do bugs happen?

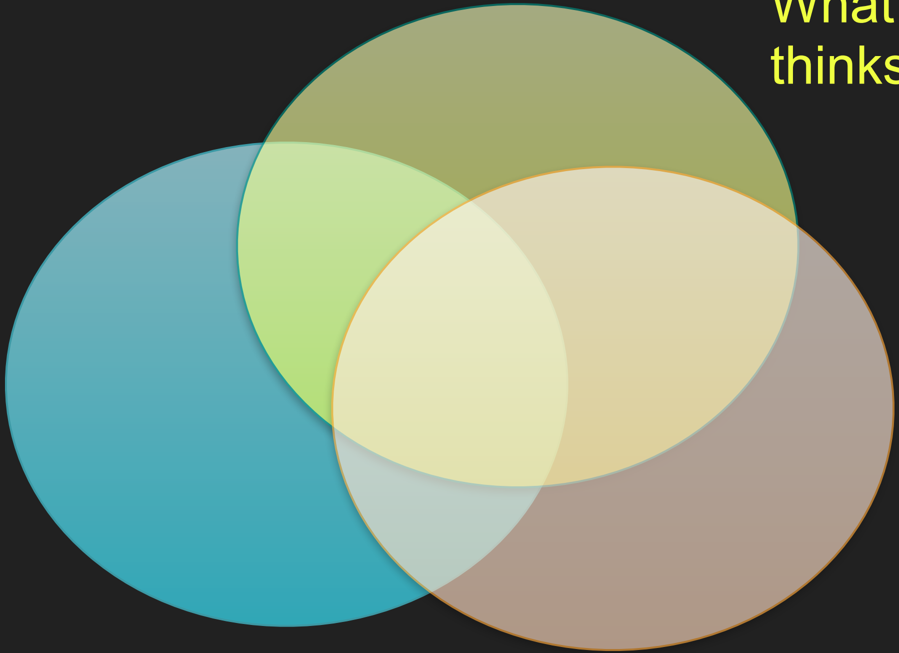
What the code should do



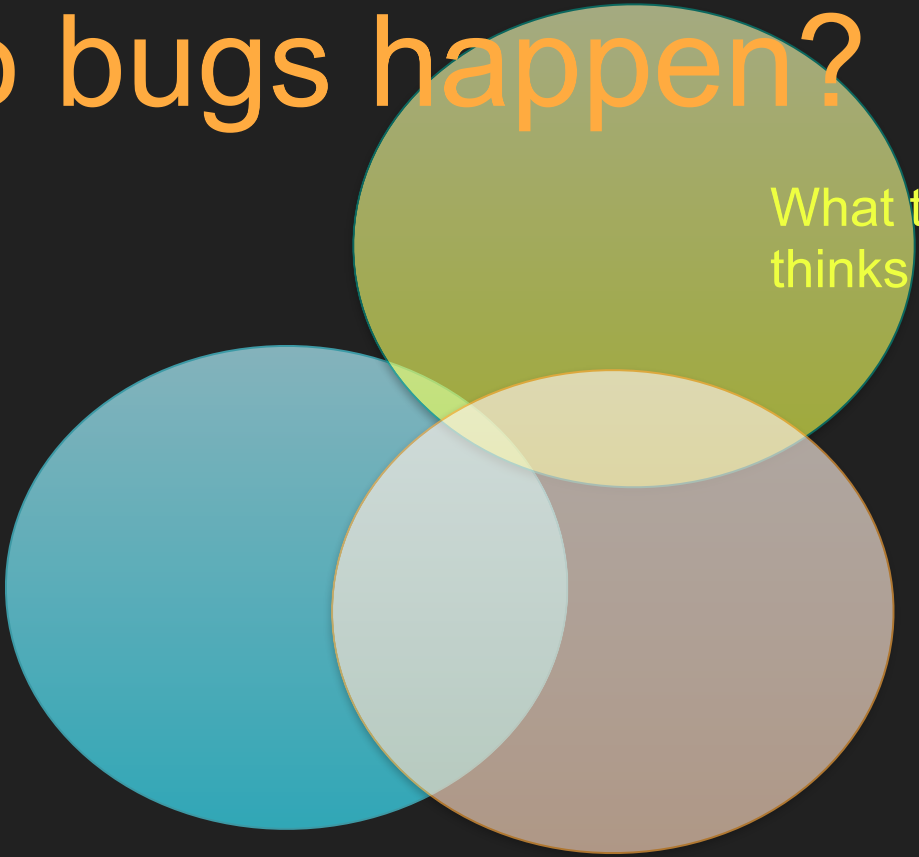
What the code actually does

@DaveLiddament

Why do bugs happen?


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

Why do bugs happen?



What the developer thinks the code does

What the code should do

What the code actually does

@DaveLiddament

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

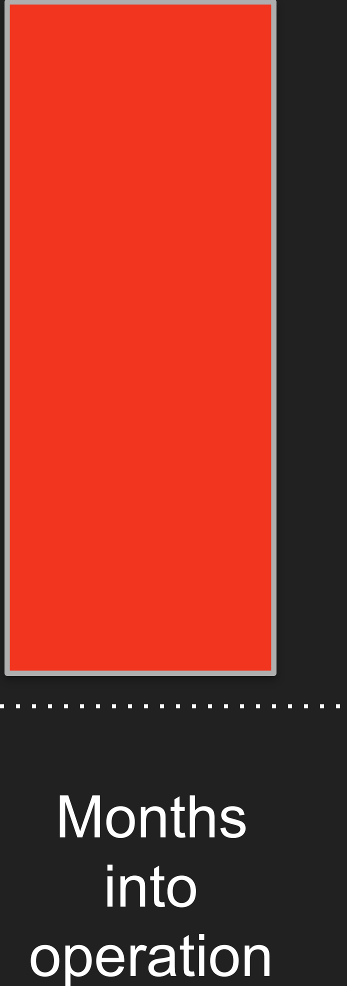
Best time to find a bug?

Best time to find a bug?

Months
into
operation

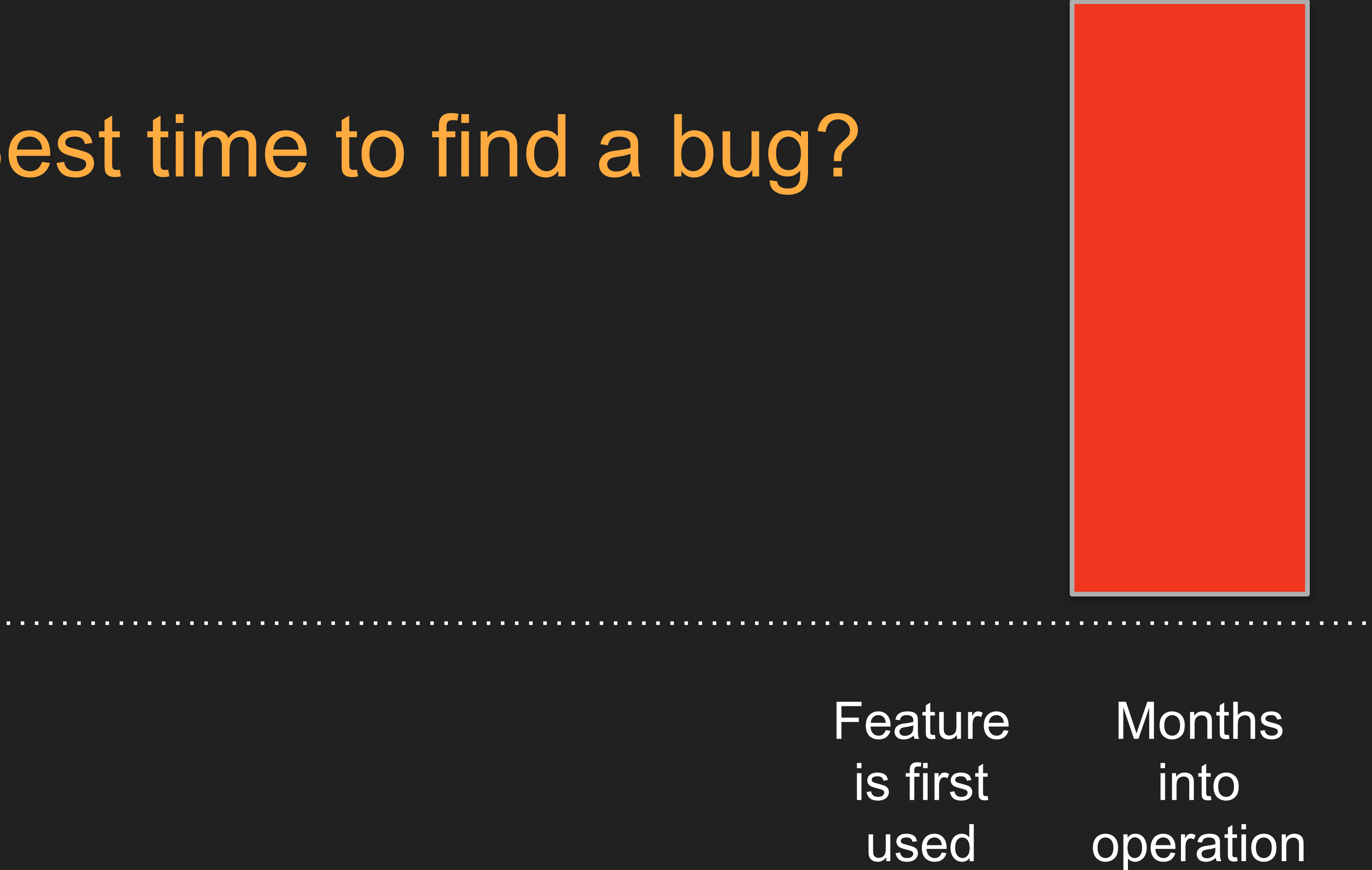
@DaveLiddament

Best time to find a bug?

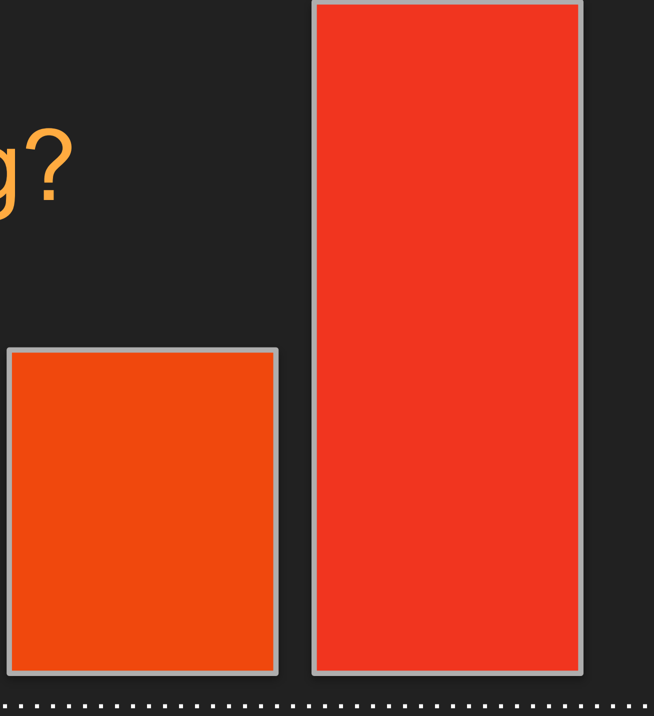


@DaveLiddament

Best time to find a bug?



Best time to find a bug?

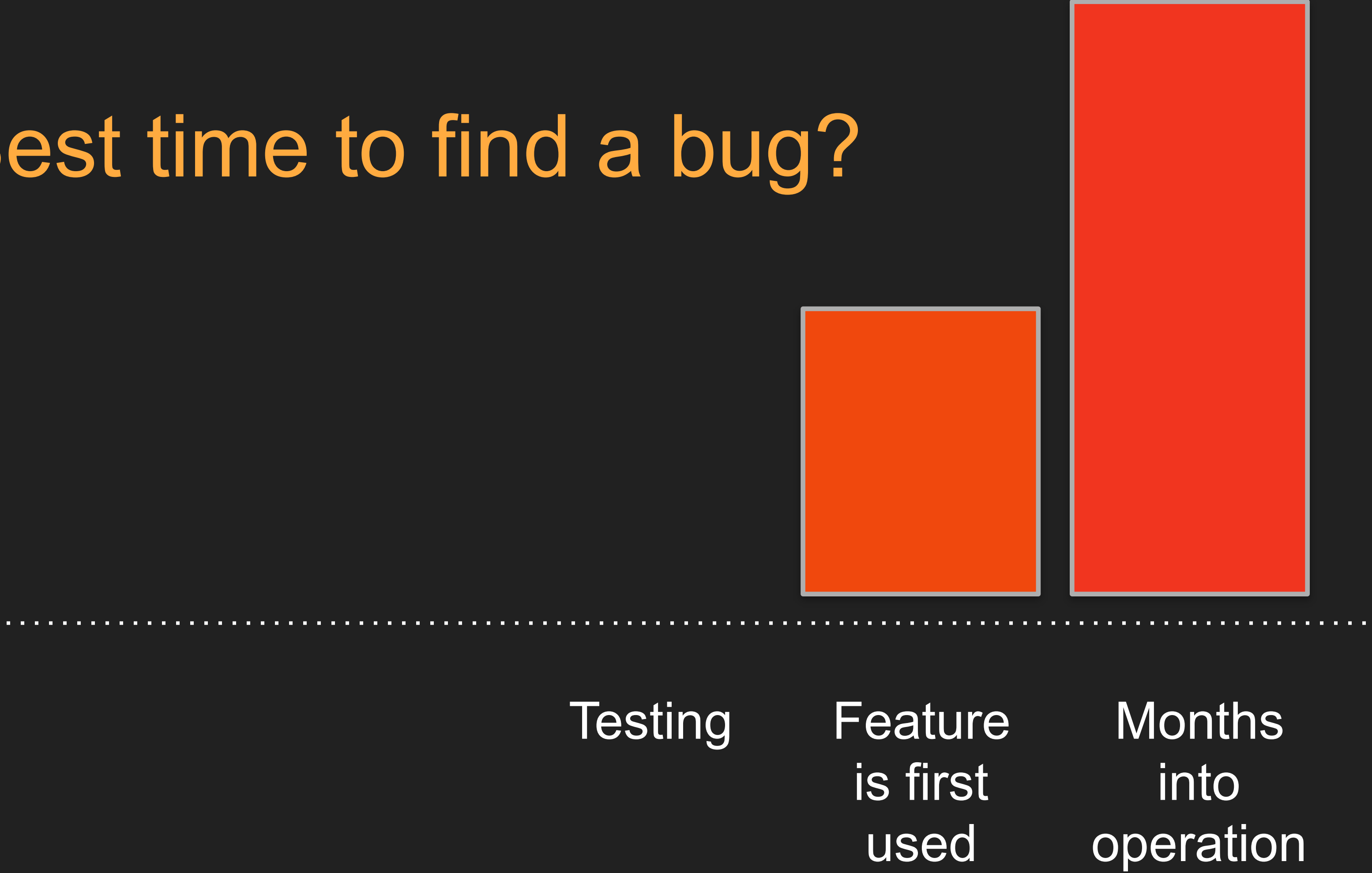


Feature is first used

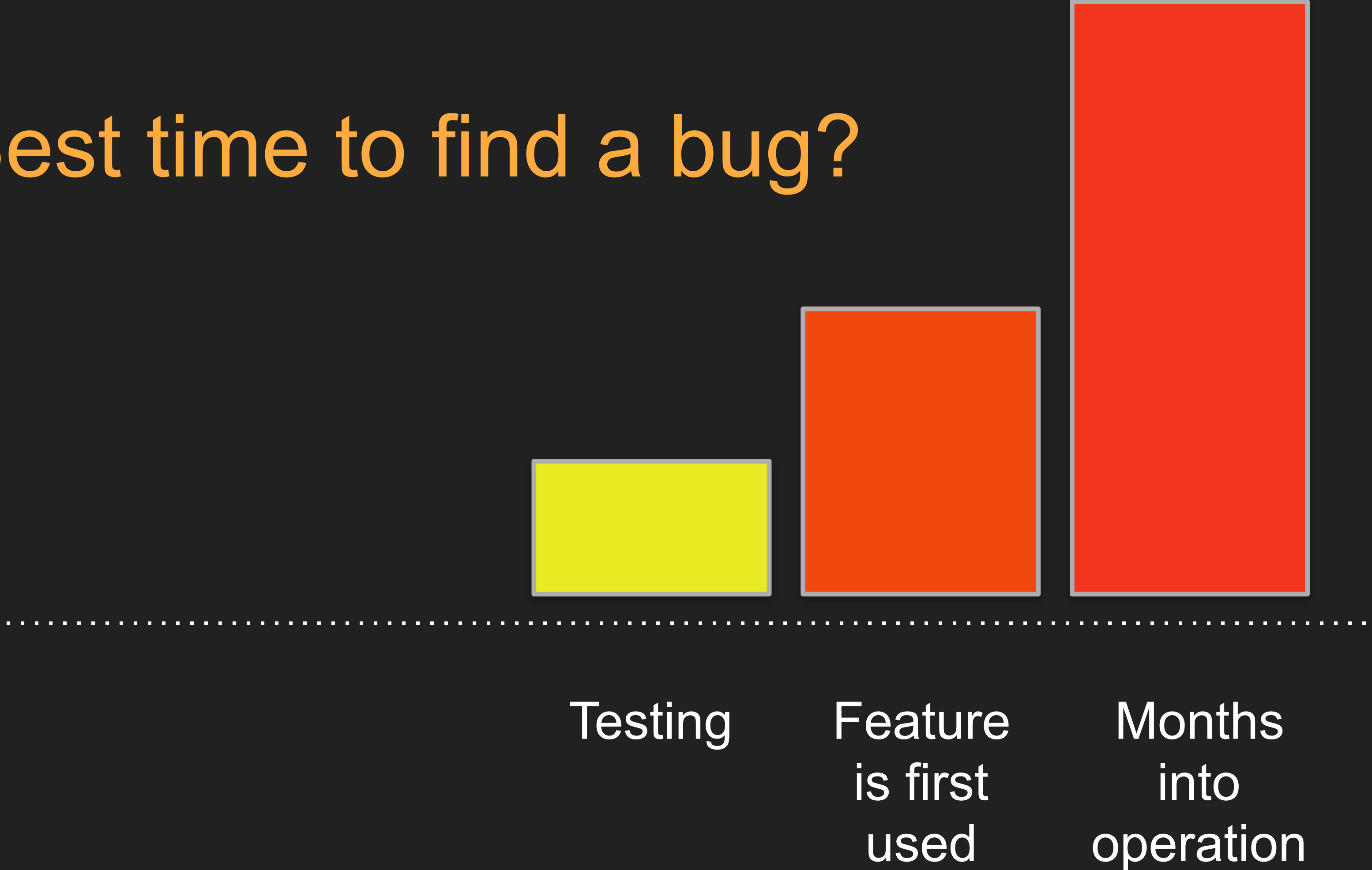
Months into operation

@DaveLiddament

Best time to find a bug?

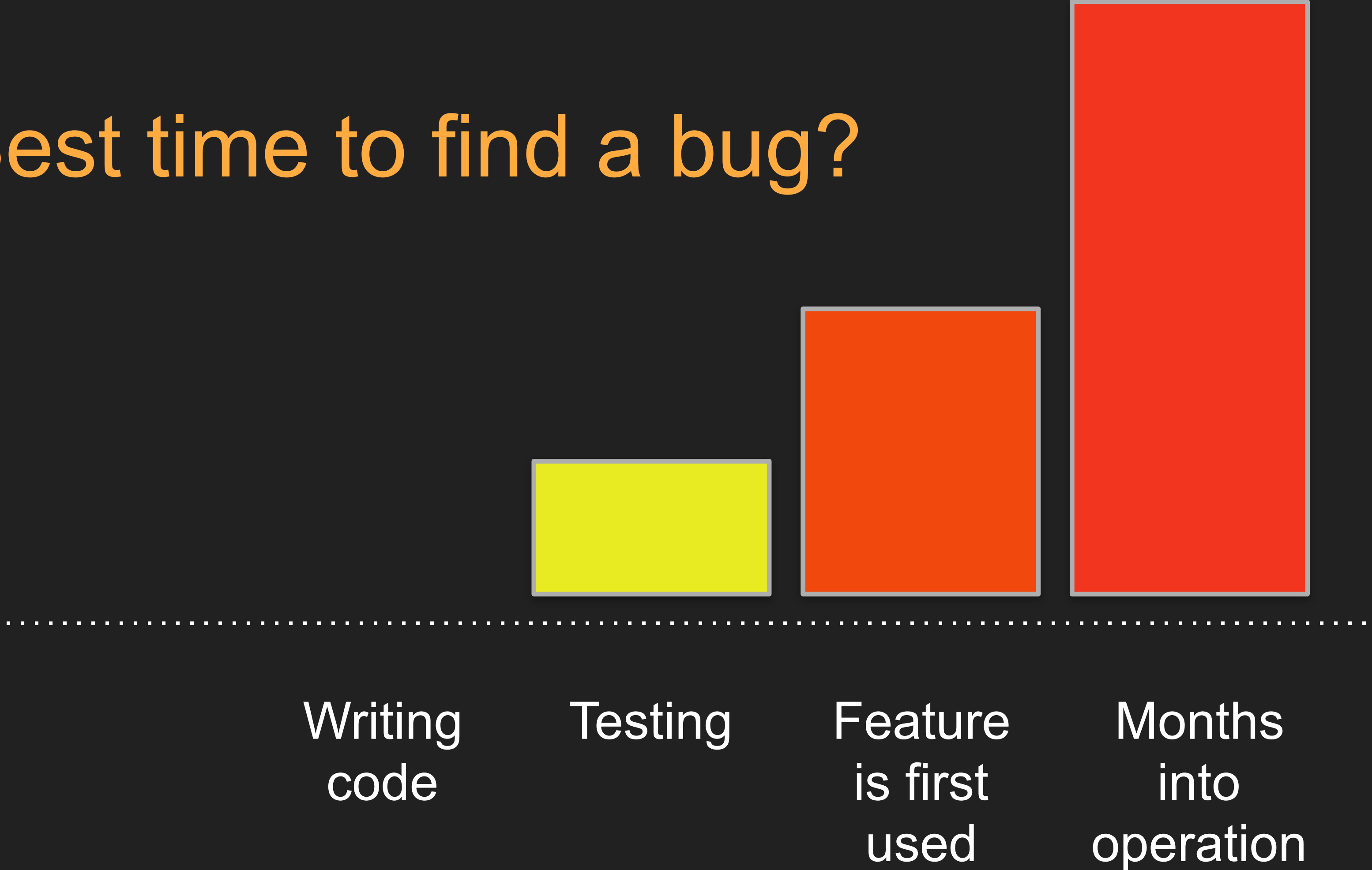


Best time to find a bug?



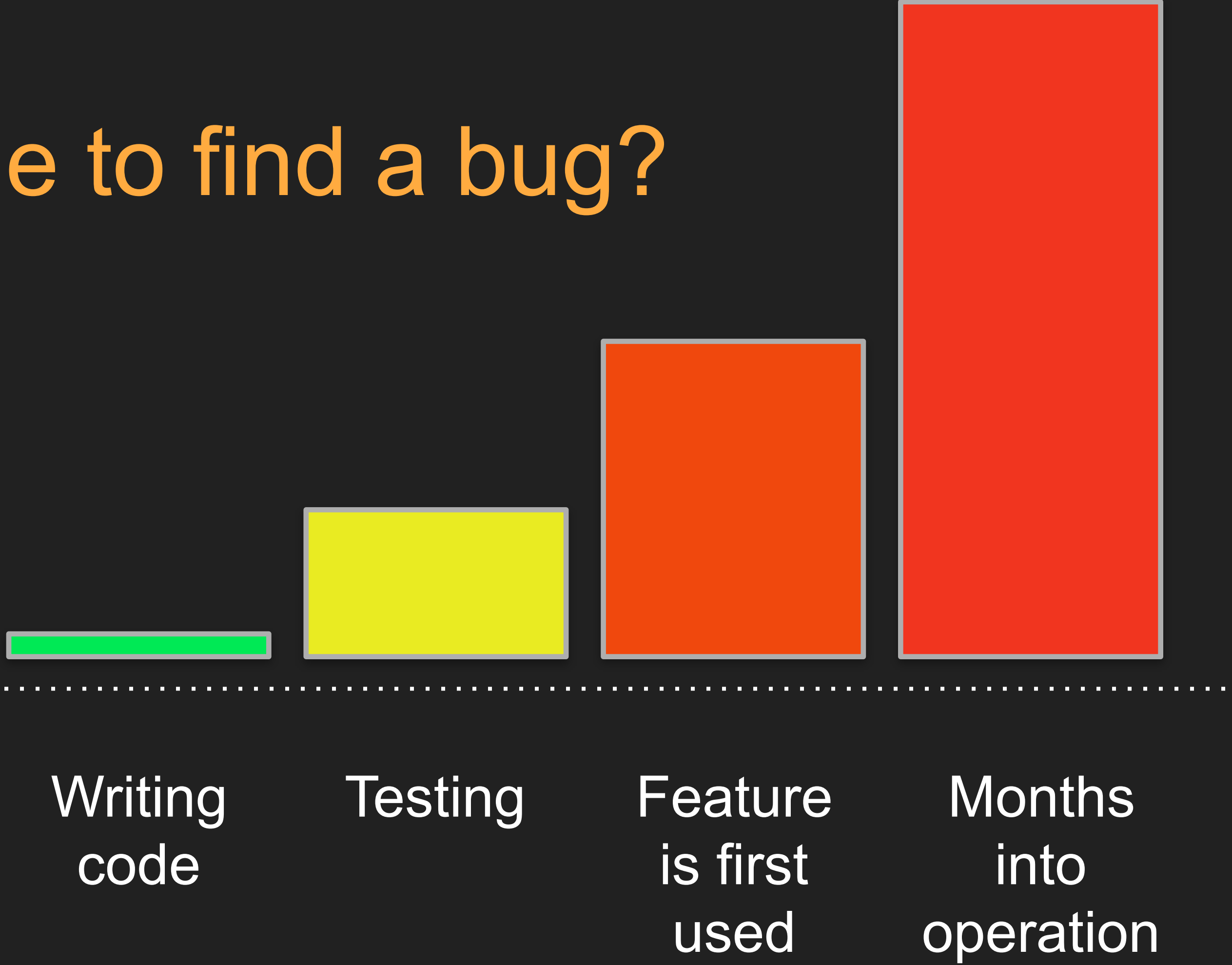
@DaveLiddament

Best time to find a bug?

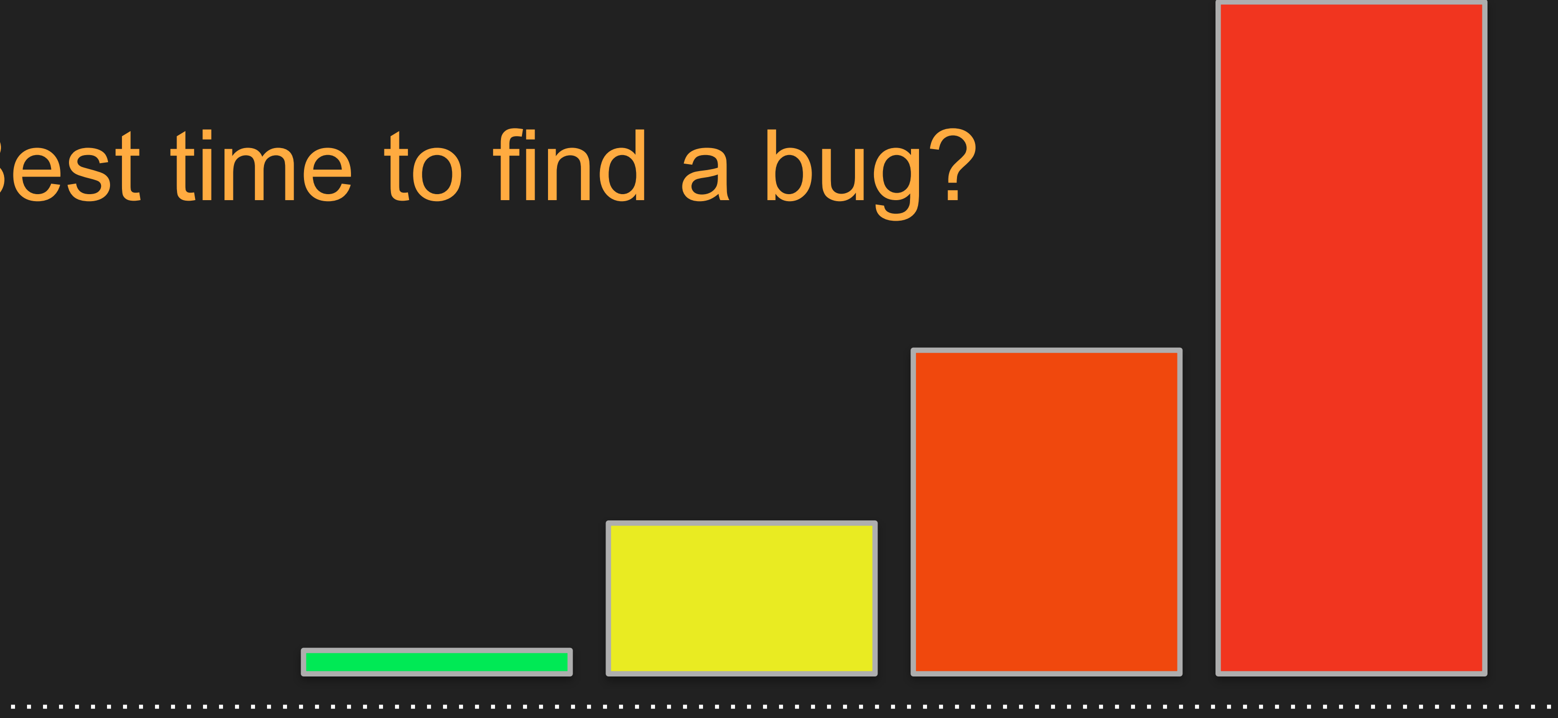


@DaveLiddament

Best time to find a bug?



Best time to find a bug?



Before writing code

Writing code

Testing

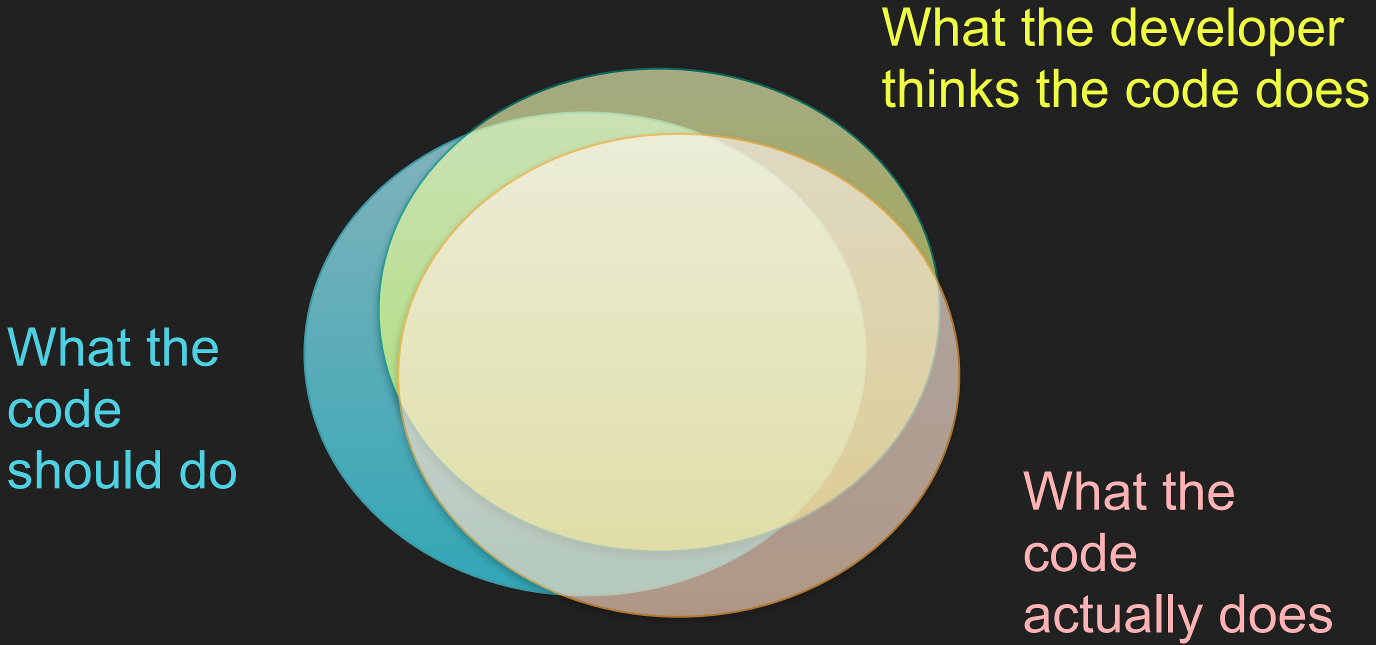
Feature is first used

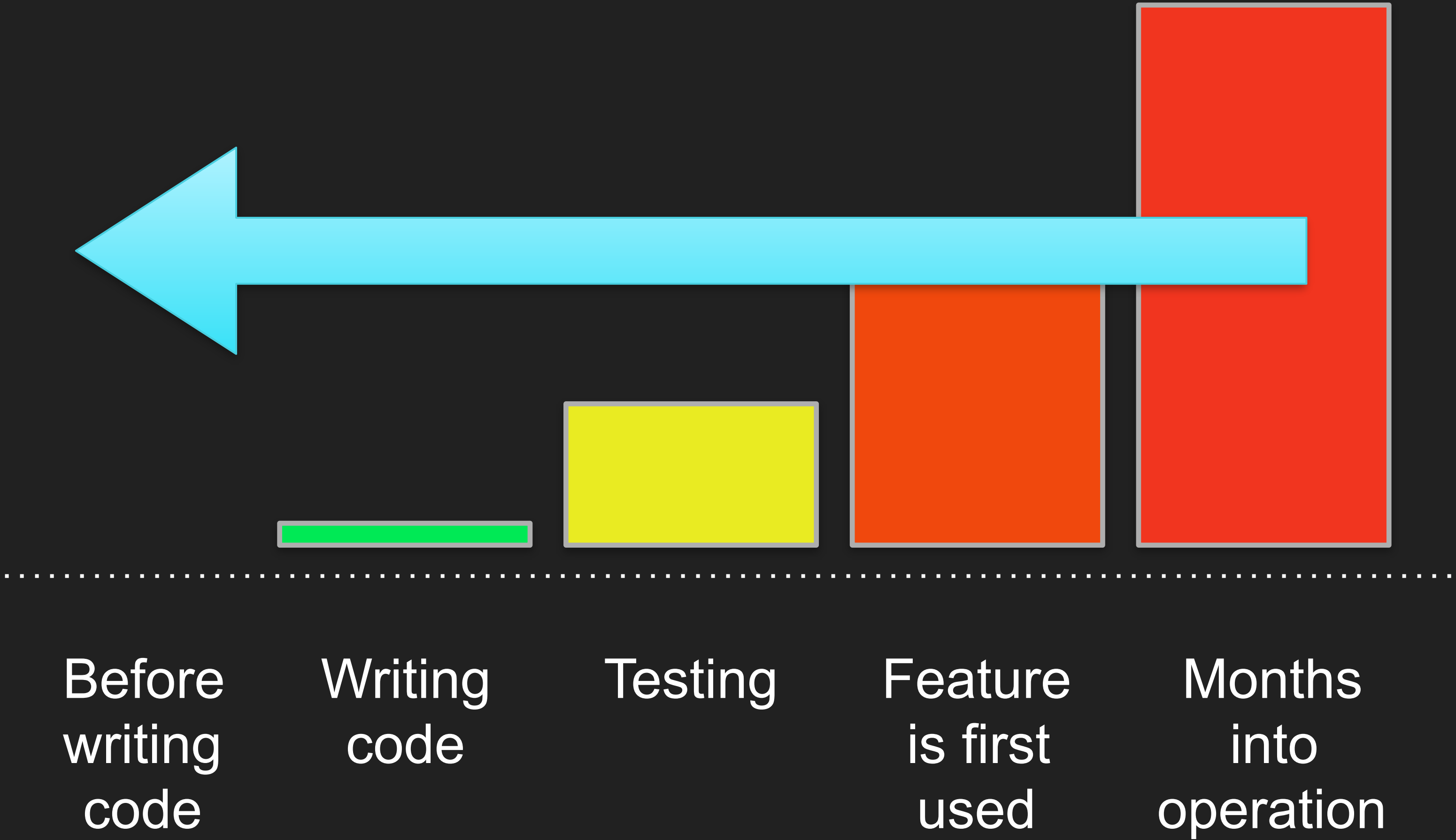
Months into operation

@DaveLiddament

Why this talk?

Improve understanding of code


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


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

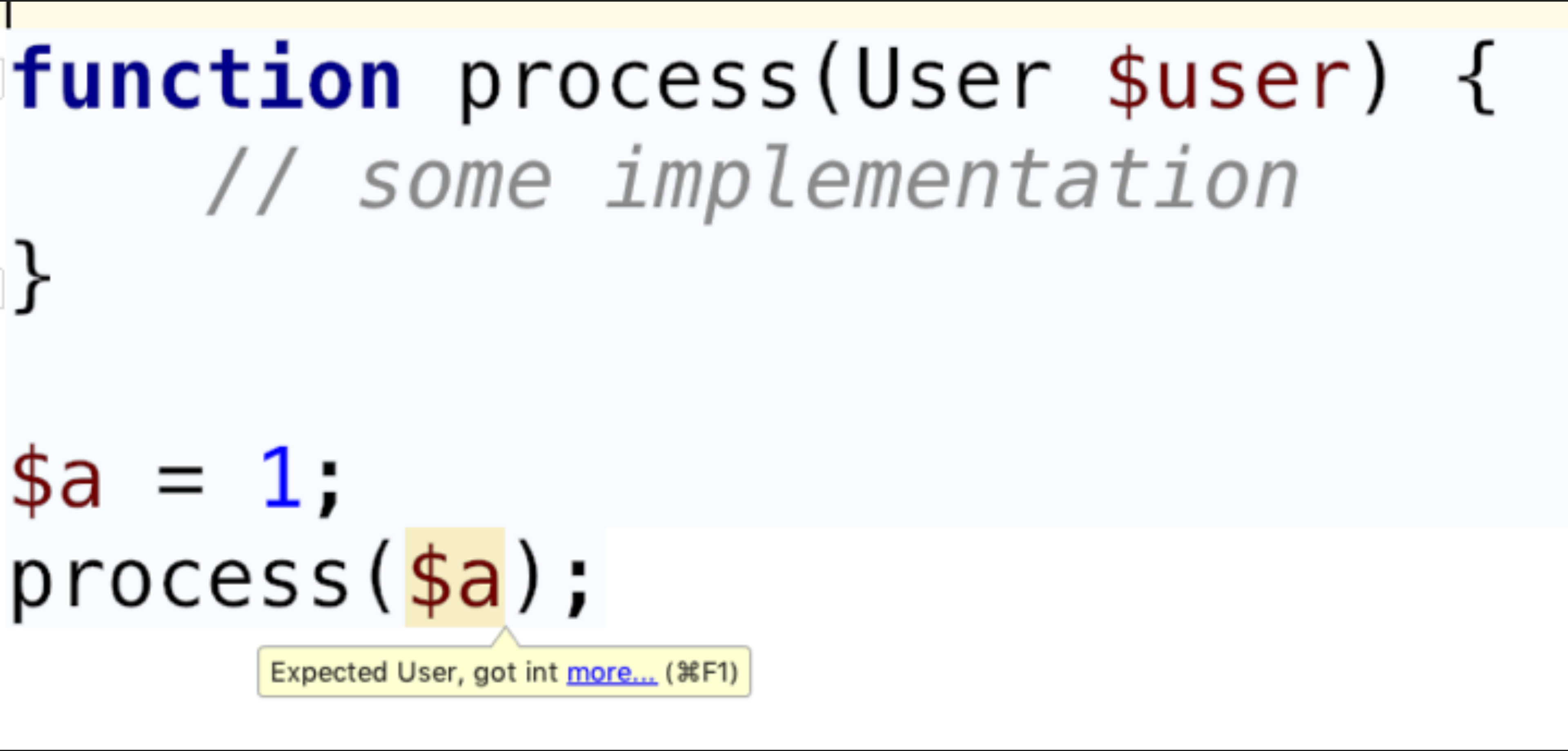
Pay attention!

Is this code valid?

```
$a = 1;

process($a);
```

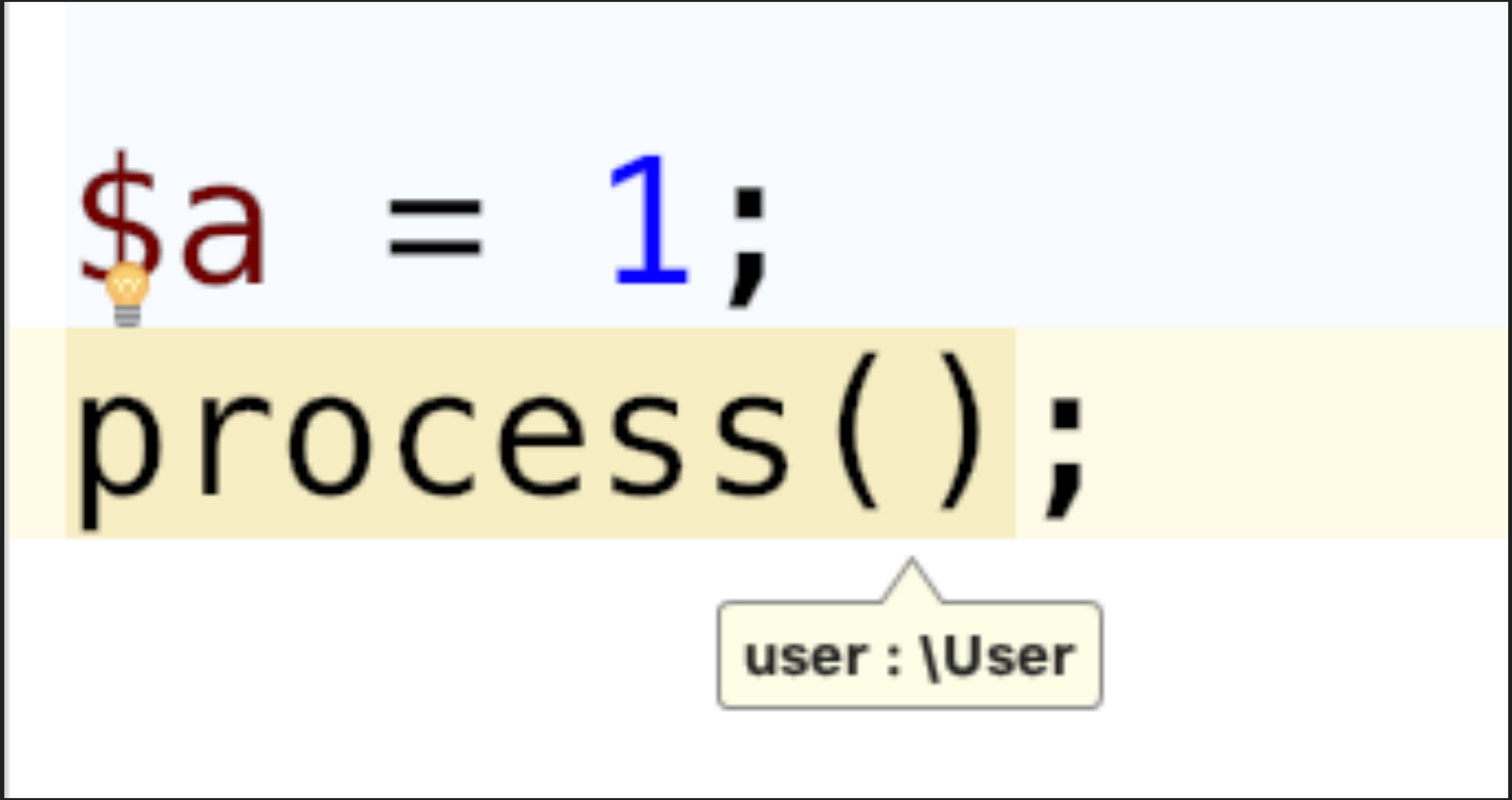
Is this code valid?

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

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



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


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

Expected User, got int more... (⌘F1)



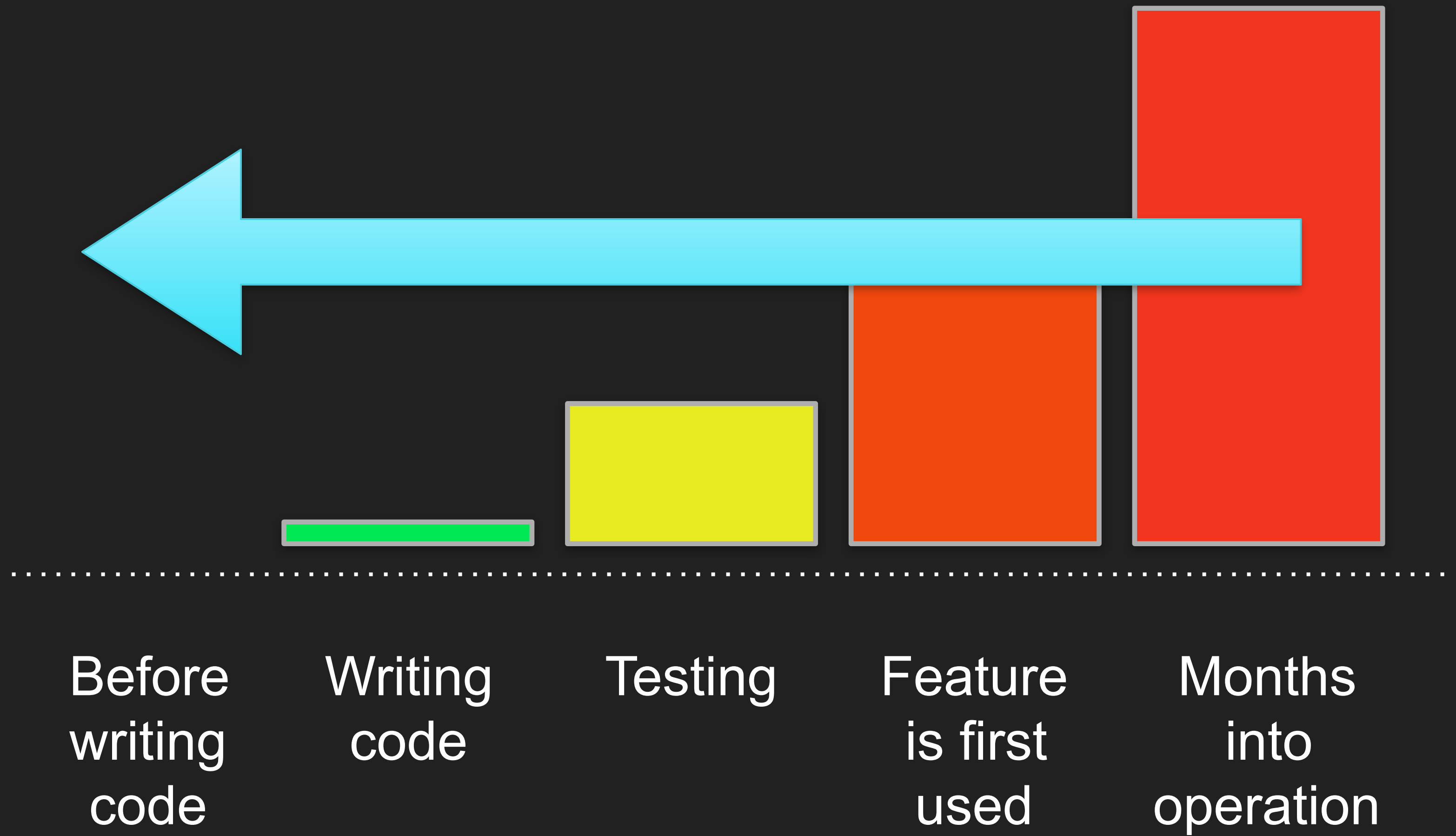
```
$a = 1;
process();
```

user : \User

Type hinting has helped

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

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

@DaveLiddament


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

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

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”)
```

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)
```

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

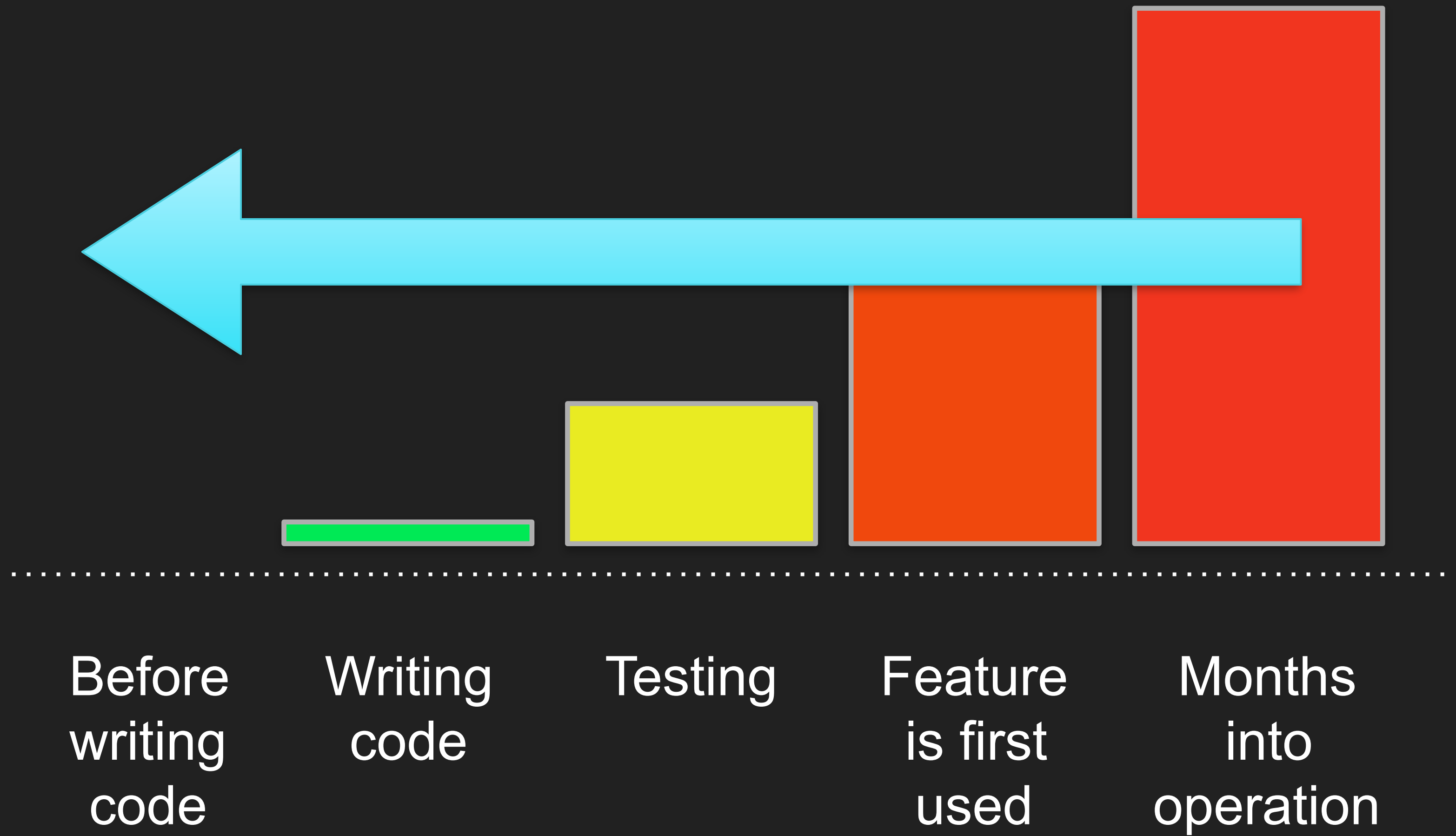
```
$emailAddress = new EmailAddress(“6 Lower Park Row”);
```

Add validation

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

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

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

@DaveLiddament

We're guaranteed that EmailAddress represents a valid email address.


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

Other example Value Objects

- Point object
- Postcode
- Settlement Type
- Status

Benefits

- Validation
- Normalise data
- Add equals method
- Add domain specific logic

Questions