Red Light ●
Green Light ●

an introduction to unit testing
and test-driven development

Objectives

- What is unit testing?
- Why should I unit test?
- What is test-driven development?
- How does TDD help me write better code?

What is unit testing?

- A method of testing discrete units of code

```php
<?php
abstract class CheckoutController {
  public function checkout() {
    // 2000 lines of code later...
  }
  public function isBml() {
    return (
      $this->_params['PaymentType'
        == 'BILLMELATER'
    );
  }
}
```

```
when I call isBml()
  … and PaymentType = BILLMELATER
    … I expect isBml() to return
      boolean(true)
  … and PaymentType != BILLMELATER
    … I expect isBml() to return
      boolean(false)
```

What is unit testing?

- Unit testing is a form of the scientific method
    - Construct a hypothesis (test case)
    - Control independent variables
    - Measure dependent variables
    - Make a conclusion (pass, fail, or skip)

What is unit testing?

```
when I call setApp($myApp)
  … I expect FooController::$_app
    to be the same as $myApp
```

```
$object = new FooController();

$myApp = new App();

$object->setApp($myApp);

$this->assertEquals(
  $myApp, // Expected value
  $object->_app // Actual value
);
```

Hypothesis

IV

DV

Conclusion

It's science!

Why should I unit test?

- Unit testing enables collective ownership

  "Who let that pass code review?!
  That would never work!"

- Unit testing prevents regressions

  "What?! I fixed that last week!
  Who broke it again?!"

Why should I unit test?

- Unit testing makes refactoring easier

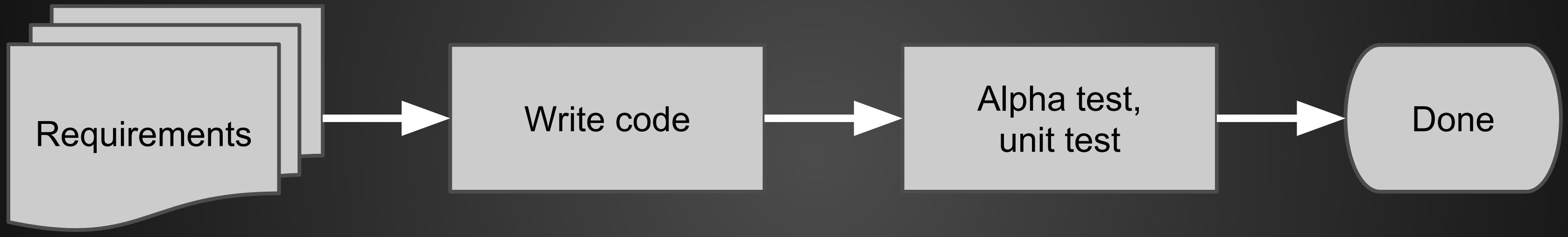
  *“1 day to refactor the code, and
  20 days to make sure it didn’t
  break everything”*

- Unit testing exposes bugs sooner

  “Well QA should have caught it!”

What is test-driven development?

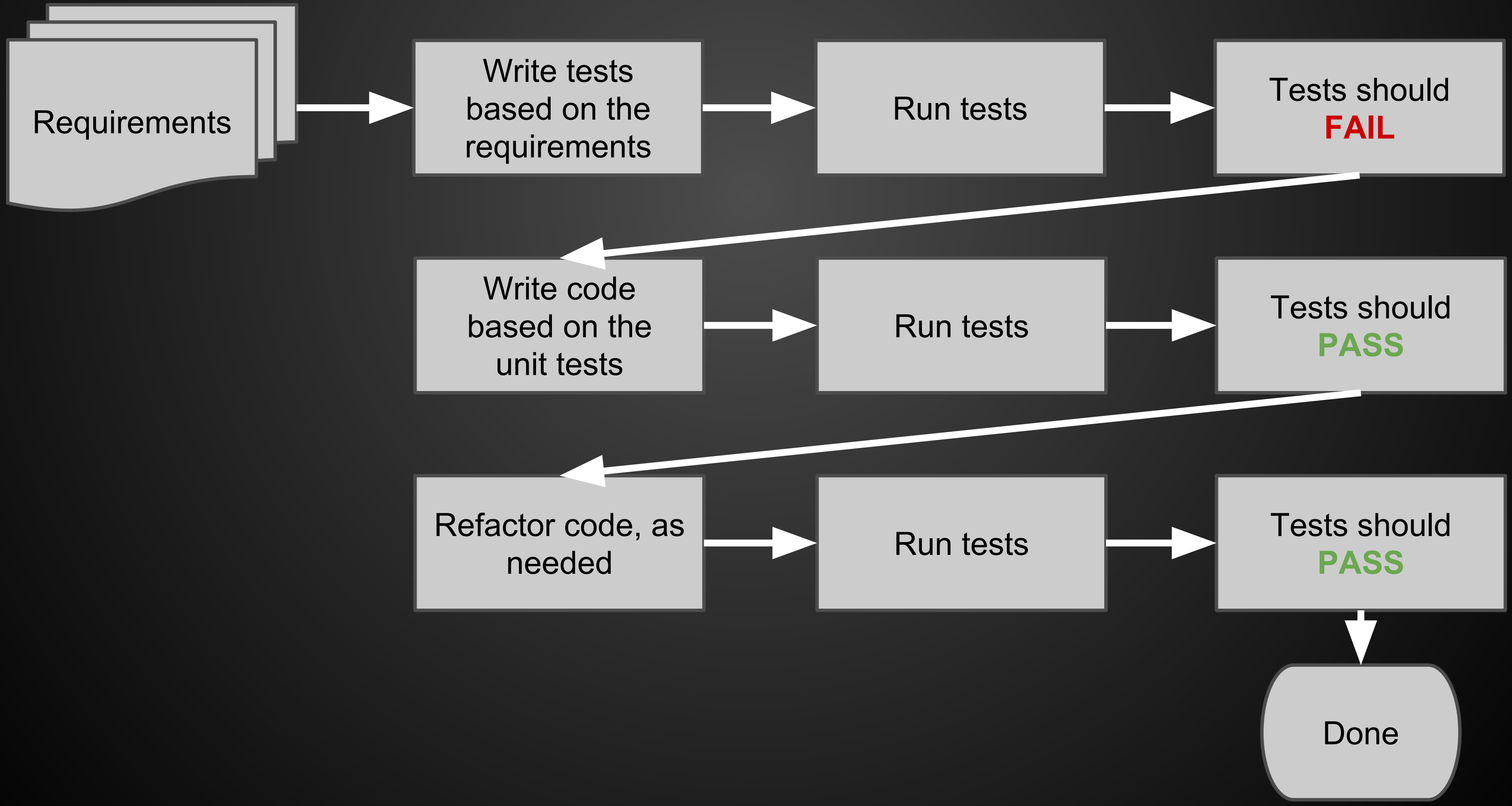
- Existing development process:



- What's the problem?
  - Our tests are skewed by confirmation bias
  - There may be edge cases we didn't predict
  - It may appear to work, but actually be failing

What is test-driven development?

- Test-driven development process:



But that’s a lot of extra work!

- It’s not as much extra work as it looks
- An ouncece of prevention is
  worth a pound of cure

  -- Benjamin Franklin

TDD leads to better code

- Of course, “better” is somewhat subjective
- Better here means:
    - Strict conformance to a narrow set of expectations
    - Lower code complexity
    - Separation of responsibilities

Strict conformance to expectations

“The function returns a negative value on error: -3 when the key length was incorrect, -4 when there was a memory allocation problem and any other return value is an unknown error. If an error occurs a warning will be displayed accordingly. FALSE is returned if incorrect parameters were passed.”

-- PHP Manual documentation for mcrypt_generic_init()

Metrics for better code

- Code Coverage
  - Percentage of logical lines of code (LLOC) executed during a test
  - Always aim for 100% code coverage, but…
  - Covering 100% of LLOC does not mean covering 100% of possible scenarios

Metrics for better code

- Cyclomatic complexity
  - The number of linearly independent paths through a piece of code
  - The number of decision points in a method, plus 1 for the method execution itself

Metrics for better code

```
    class Foo {
1     public function doSomething($a, $b) {
2       if ($a < 0) {
          $a *= -2;
3       } elseif ($a > 0) {
4         if ($a % 2 == 1)
            $a *= 2;
        } else {
        }
5       if ($b > 0) {
6         while ($b % 18 != 0) {
            $b += 1;
          }
        }
        return $a + $b;
      }
    }
```

Metrics for better code

- N-Path complexity
  - The number of acyclic execution paths for a method
  - Where cyclomatic complexity is the number of independent decision points, n-path complexity is the number of permutations of these decision points

Metrics for better code

```
    class Foo {
1     public function doSomething($a, $b) {
2       if ($a < 0) {
          $a *= -2;
3       } elseif ($a > 0) {
4         if ($a % 2 == 1)
            $a *= 2;
        } else {
        }
5       if ($b > 0) {
6         while ($b % 18 != 0) {
            $b += 1;
          }
        }
        return $a + $b;
      }
    }
```

2, 5, 6

2, 5, ¬6

2, ¬5

3, 4, 5, 6

3, 4, 5, ¬6

3, 4, ¬5

3, ¬4, 5, 6

3, ¬4, 5, ¬6

3, ¬4, ¬5

¬2, 5, 6

¬2, 5, ¬6

¬2, ¬5

Separation of responsibilities

- "What does this class do?"
  - Should a model class execute database queries?
  - Or should a model class use Mysqli to execute database queries?

- When testing a class, you want to test:
  - 100% of the LLOC in the class
  - Not a single line outside of the class
  - This is only possible through separation of responsibilities

Obstacles to unit testing

- Reliance on the database
- Visibility issues
- Global scope
    - This means you, Singleton Pattern!
    - DatabaseConnection::getInstance()
- Untestable code
    - Calls to native PHP functions
    - Object instantiation
    - Use of parent:: and self::
    - Certain uses of private methods, properties

You’ve learned...

- What unit testing is
- Why you should unit test
- What test-driven development is
- How TDD will help you write better code

The End

Thank you!

About the Author

Web Developer at BookIt.com

GitHub:
github.com/matthewpatterson

LinkedIn:
linkedin.com/in/matthewspatterson

Twitter:
@mattpattwhereat

Special Thanks

Donald Hulce, Ryan Null, Vladimir Loscutoff, and Iain Saxton

Bidgee for *LED traffic light in Forest Hill, New South Wales.* (Slide 1)

- ➔ Original Image: http://upload.wikimedia.org/wikipedia/commons/4/47/LED_traffic_light.jpg