Test Driving Javascript

•••

Jessie Puls - jpuls@pillartechnology.com
Github - https://github.com/jessiepuls
Twitter - @jessiepuls

Today's Agenda

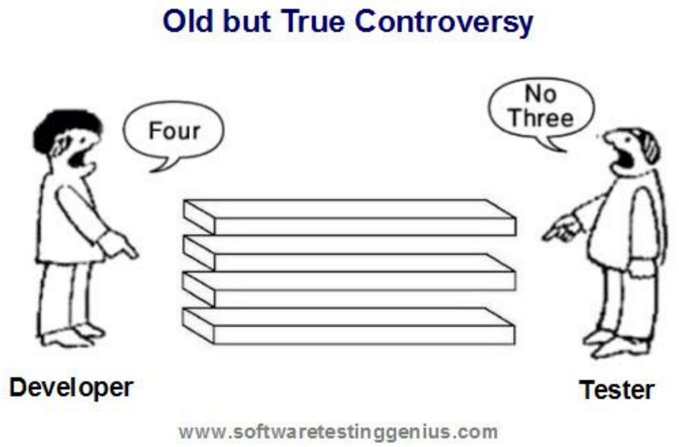
- ➔ General overview
- ➔ Get into groups & set up
- ➔ Exercise 1: FizzBuzz
- ➔ Exercise 2: Sum of digits
- ➔ Exercise 3: Magic 8-ball
- ➔ Extra Credit: Something of your own design

What we'll be using

- ➔ Jasmine as our testing framework
- ➔ Karma as our test runner
- ➔ JQuery because it's easy to get up and running

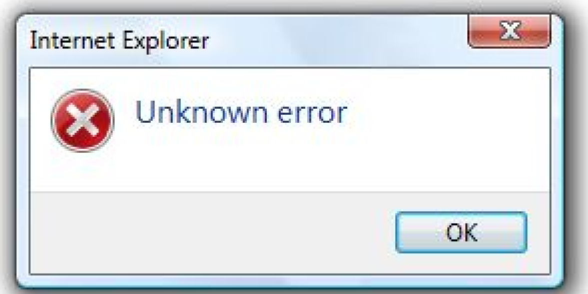
Jasmine: how Do I write tests?

```
describe("Brief description of our test suite", function () {
    beforeEach(function() {
        // set up before each test
    });

    afterEach(function() {
       // clean something up after each test
    });

    it("Describe the thing you want to test", function () {
        expect(true).toBe(true);
    });
});
```



Jasmine: How do I test errors?

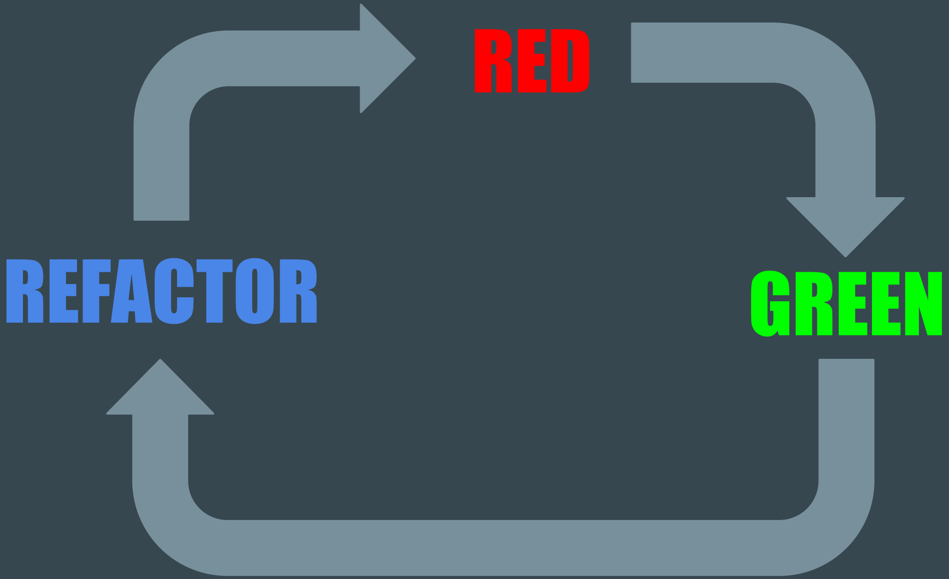
```
describe("Brief description of our test suite", function () {
    ...
    it('should throw an exception if not passed a parameter', function () {
        expect(function(){ someFunction(); })
            .toThrow(new Error('Invalid Argument'));
     });
});
```



Internet Explorer

Unknown error

GREAT, SO HOW DO I TEST DRIVE MY CODE?



Ping Pong Pairing

- ➔ I write a test
- ➔ You make it pass
- ➔ Refactor/commit
- ➔ You write a test
- ➔ I make it pass
- ➔ Refactor/Commit

Example: Palindromes

Palindromes are words that are the same forward as they are backward.

Example: madam

Given a string

1. If string == gnirts return true
2. If string != gnirts return false
3. If string is not really a string throw an error
4. Ignore whitespace while comparing
5. Ignore capitalization while comparing

Our first test

```
it('tacocat is a palindrome', function () {
    expect(isPalindrome('tacocat')).toBe(true);
});
```

..and make it pass in the easiest way

```
var isPalindrome = function(inputString) {
    return true;
};
```

Tricky...Let’s add another test

```
it('tacodog is not a palindrome', function () {
    expect(isPalindrome('tacodog')).toBe(false);
});
```

And then write some real code

```
var isPalindrome = function(inputString) {
    var reverseString = inputString.split('').reverse().join('');
    return reverseString == inputString;
};
```

How about that error case

```
it('should throw an exception if not passed a string', function () {
    expect(function(){ isPalindrome(123); })
        .toThrow(new Error('Invalid Argument'));
});
```

And the code

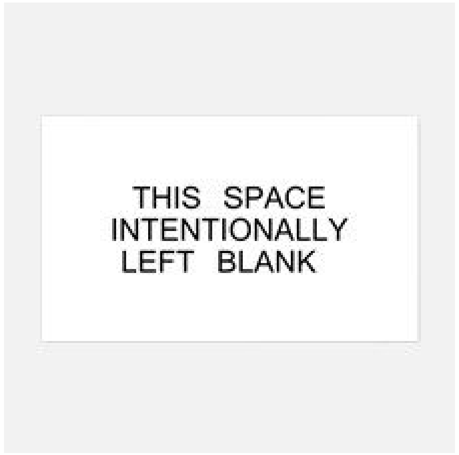
```
var isPalindrome = function(inputString) {
    if(typeof inputString != 'string') {
        throw new Error('Invalid Argument');
    }

    var reverseString = inputString.split('').reverse().join('');
    return reverseString == inputString;
};
```

Ignore whitespace

```
it('should ignore whitespace', function() {
    expect(isPalindrome('taco cat')).toBe(true);
});
```

More code!

```
var isPalindrome = function(inputString) {
    if(inputString == undefined) {
        throw new Error('Invalid Argument');
    }

    var reverseString = inputString.split('').reverse().join('').replace(' ', '');
    return reverseString == inputString.replace(' ','');
};
```


THIS SPACE
INTENTIONALLY
LEFT BLANK

This is getting a little ugly. Let’s try a refactor

```
var isPalindrome = function(inputString) {
    if(inputString == undefined) {
        throw new Error('Invalid Argument');
    }

    var reverseString = inputString.split('').reverse().join('').replace(' ', '');
    return reverseString == inputString.replace(' ','');
};
```

This is a bit more readable, and our tests still pass!

```
String.prototype.reverse = function() {
    return this.split('').reverse().join('');
};

String.prototype.removeWhitespace = function() {
    return this.replace(' ', '')
};

String.prototype.isPalindrome = function() {
    return this.removeWhitespace().reverse() == this.removeWhitespace();
};

var isPalindrome = function(inputString) {
    if(inputString == undefined) {
        throw new Error('Invalid Argument');
    }

    return inputString.isPalindrome()
};
```

Okay, let’s keep going. What about tabs & returns?

```
it('should ignore whitespace', function() {
    expect(isPalindrome("taco \t\r\ncat")).toBe(true);
});
```

And now we just have to update one place

```
String.prototype.removeWhitespace = function() {
    // remember this used to be this:
    // return this.replace(' ', '')
    return this.replace(/\s/g, '')
};
```

Last case...ignore capitalization

```
it('should ignore capitalization', function() {
    expect(isPalindrome("Taco cat")).toBe(true);
});
```

And we update our comparison

```
String.prototype.equalsIgnoreCase = function(str) {
    return this.toLocaleLowerCase() == str.toLowerCase();
};

String.prototype.isPalindrome = function() {
    return this.removeWhitespace()
               .reverse()
               .equalsIgnoreCase(this.removeWhitespace());
};
```

Let’s get started!

- ➔ Groups of 2-4 people
- ➔ Grab a USB drive
- ➔ Follow the directions in README.md
- ➔ Get started on exercise one

Exercise One: Fizz Buzz

Given a number:

1. If it is divisible by 3 output Fizz
2. If it is divisible by 5 output Buzz
3. If it is divisible by 3 and 5 output Fizz Buzz
4. Otherwise output the number

So inputs 1, 2, 3, 4, 5, 15

Create outputs 1, 2, Fizz, 4, Buzz, Fizz Buzz

Exercise Two: Sum of Digits

Given a number

1. Return the sum of the digits in that number
2. If non numeric values are input throw an error

Inputs: 1, 12, MOM

Should produce outputs: 1, 3, Invalid Input

How’d it go?

…

Pulling data from a REST service

- ➔ We'll use jasmine-ajax to mock our endpoints, so tests can run frequently without impacting, or relying on outside services
- ➔ See exampleEndpointMocking.spec.js for reference

Example: Setup

```
describe('Example endpoint test', function () {
    beforeEach(function () {
        jasmine.Ajax.install();
    });

    afterEach(function () {
        jasmine.Ajax.uninstall();
    });

    it('send a GET request to the correct endpoint', function () {
        ws.exampleEndpointCall('something');
        expect(jasmine.Ajax.requests.mostRecent().method).toBe('GET');
        expect(jasmine.Ajax.requests.mostRecent().url).toBe(ws.endpoint +
'something');
    });
});
```

This is setup that has to be done for any tests where we want to intercept ajax calls

Example: Faking Responses

```
describe('Successful requests', function () {
    it("should set the value on success", function() {
        spyOn(ws, 'setValue');
        ws.exampleEndpointCall('something');

        jasmine.Ajax.requests.mostRecent().respondWith({
            'status': 200,
            'contentType': 'application/json',
            'responseText': '{"data": {"someProperty": "someValue"}}'
        });

        expect(ws.setValue).toHaveBeenCalledWith('someValue');
    });
});
```

Since ajax requests are asynchronous we can send the request after the response has been made

Exercise Three: Magic Eight Ball

Create a webpage that will answer questions with random responses pulled from the rest enpoint provided by https://8ball.delegator.com/

1. Create a webpage where a user can ask a question that will be answered by the magic 8-ball service.
2. Make sure you mock the endpoint rather than actually hitting it while you run your tests.
3. Visually indicate in some way or another if you’ve gotten back an “affirmative” or “contrary” answer.

Exercise Four: Come up with your own idea

Do something fun. Here are a few sources with APIs to get your started, but feel free to do anything.

1. Star Wars API - https://swapi.co/documentation
2. Marvel API - http://developer.marvel.com/documentation/getting_started
3. Brewery DB - http://www.brewerydb.com/developers
4. Twitter - https://dev.twitter.com/rest/public

Questions?

Jessie Puls - jpuls@pillartechnology.com
Github - https://github.com/jessiepuls
Twitter - @jessiepuls

The End