

UNIT TESTING

Ana Rita Silva, Manuel Zamith

TVVS

Friday, October 7, 2016

What is this?

Unit Testing Techniques

- Black Box Testing:** Testing a system without having specific knowledge to the internal technology of the system, no actions to the source code, and no knowledge of the variables.
- White/Clear Box Testing:** Testing a system with full knowledge and access to all source code and architecture documents.
- Grey Box Testing:** Testing a system while having at least some knowledge of the internals of a system.

Why is this used?

- Tests have specific behavior
- Improves efficiency and reliability
- Can verify the quality of a system - the unit test itself is used to verify the design
- You can be more confident about your code
- Each test is autonomous

Benefits

- Improves efficiency and reliability
- Tested areas in many thousands times, so the whole system is tested
- Knowledge of testing - which are included in it: requirements, design and actions, better refactoring, and integration testing is easier

Limitations

- Time consuming
- Only tests what is explicitly coded
- Intelligence of system being tested can be tested
- Value and usefulness of Unit tests can be often debated - what and when to write them, how much time to spend on them, etc.
- Does not help to detect architectural requirements - can lead to a false sense of security for the system being tested

Unit Testing Tools

C/C++

- VECTOR
- CUTE++
- CPPUTEST

Javascript

- Jasmine
- Karma
- Chai
- Mocha
- QUnit

.NET & others

- Microsoft Test Manager
- Visual Studio Test
- NUnit
- SpecFlow
- SpecFlow+Runner

Code Coverage Tools for Javascript

- JSCover - free and coverage measured via a browser
- BLANKETJS - deprecated for jasmine version
- jasmine-istanbul - embedded in KARMA
- KARMA - spawns a web server that executes source code against test code for each of the browsers connected

Using Jasmine & KARMA

Codebase

- Specs:** Functional test that contains 1 or + expectations
- Expectations:** Assertion that something IS
- String:** like the spec

Jasmine Setup and TearDown

- Setup:** Function to run before each spec
- AfterEach:** Function called once after each spec
- Describe:** Which function it called

Mode & status

- Jasmine**: jasmine.getEnv().createTestEnv();
- Mode:** **Specs** **Setup** **AfterEach** **Describe**

Notes: Code to describe can be injected, with specs defined at any level - type of function connection. Before a spec is executed, Jasmine walks down the tree, executing each **describe** function in order.

Exercise

Can you find the errors in source code?
Achieve 100% coverage and add new test case(s).

File: test.js

```
describe('My first test', function() {
    it('should pass', function() {
        expect(true).toBe(true);
    });
});
```

Please follow the instructions provided in:
<http://www.karma-runner.org/getting-started.html>

References

- http://www.martinfowler.com/bliki/UnitTesting.html
- http://www.tutorialspoint.com/unit_testing/index.htm
- http://www.guru99.com/unit-testing.html
- http://www.theserverside.com/tutorials/Java-Server-Frameworks/Unit-Testing-with-JUnit
- http://www.theserverside.com/tutorials/Java-Server-Frameworks/Unit-Testing-with-Maven
- http://www.theserverside.com/tutorials/Java-Server-Frameworks/Unit-Testing-with-Surefire

“Fools figure complexity. Pragmatists suffer it. Some can avoid it. Geniuses remove it.”
Alan J. Perlis

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

UNIT TESTING

Ana Rita Silva, Manuel Zamith

TVVS

Friday, October 7, 2016

What is this?

Unit Testing Techniques

- Black Box Testing:** Testing a system without having specific knowledge to the internal technology of the system, no actions to the source code, and no knowledge of the variables.
- White/Clear Box Testing:** Testing a system with full knowledge and access to all source code and architecture documents.
- Grey Box Testing:** Testing a system while having at least some knowledge of the internals of a system.

Why is this used?

- Tests one specific behavior
- Improves efficiency and reliability
- Can verify the whole system - the unit test itself is used to verify the design
- You can be more confident about your code
- Each test is autonomous

Benefits

- Improves efficiency and reliability
- Reduces defects in early development phases by catching them earlier
- Knowledge of testing - which are collected in a report
- Improves design and avoids better refactoring of code
- Integrates testing in code

Limitations

- Time consuming
- Only tests the external interface
- Independent series of points testing can be tested
- Value and execution of unit tests can be often related to initial requirements
- Does not help to detect architectural requirements - can lead to a false sense of security for the testing phase, one testing code that covers the test

Unit Testing Tools

C/C++

Javascript

.NET & others

Code Coverage Tools for Javascript

- JSCover - free coverage measured via a browser
- BLANKETJS - deprecated for jasmine version
- jasmine-istanbul - embedded in KARMA
- KARMA - spawns a web server that executes source code against test code for each of the browsers connected

Using Jasmine & KARMA

Codebase

- Specs:** Functional test that contains 1 or + expectations
- Expectations:** Assertion that something is true
- String:** like the spec

Jasmine Setup and TestRun

- Specs:** A file containing the specs before each spec
- afterEach:** Function called once after each spec
- describe:** Which function it called

Mode & status

Browser: jasmine.getEnv().currentSpec().status();

Notes: Code to describe can be injected, with specs defined at any level - type of function connection. Before a spec is executed, Jasmine walks down the tree, executing each describe function in order.

Exercise

Can you find the errors in source code?
Achieve 100% coverage and add new test case(s).

File: spec/test.js

Pass 0% | Pending 0% | Failed 0% | Errors 0% | Skipped 0% | Total 0% | Coverage 0%

Please follow the instructions provided in:
<http://www.karma-runner.org/config.html>

References

“ Finds figure complexity. Pragmatists suffer it. Some can avoid it. Geniuses remove it. Alan J. Perlis

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

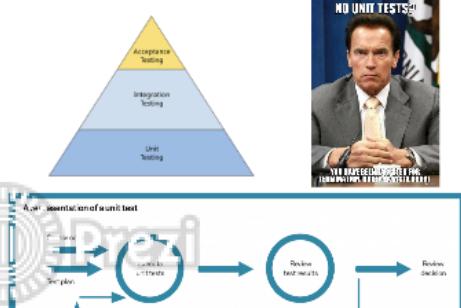
UNIT TESTING

Ana Rita Silva, Manuel Zamith

TVVS

Friday, October 7, 2016

What is this?



Why is this used?

- Tests one specific behavior
- Improves efficiency and reliability
- Can act as documentation of the system - the unit test itself is used to verify the design
- You can be more confident about your code
- Each test is autonomous

Benefits

- Improves efficiency and reliability
- Reduces defects in newly developed features or reduces bugs when changing the existing functionality
- Reduces cost of testing - defects are detected in an earlier stage

Unit Testing Tools

C/C++



PHP



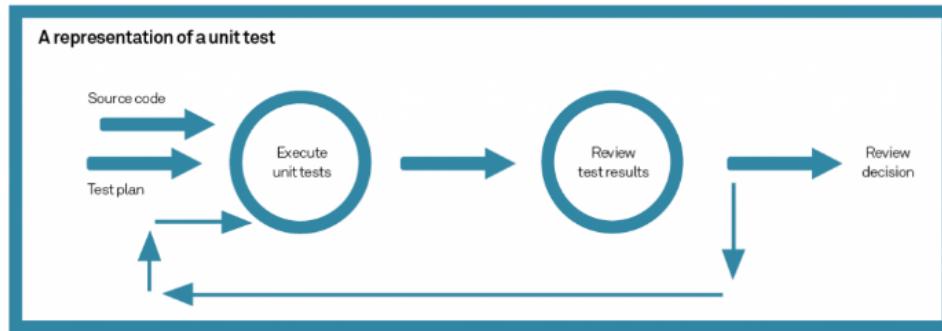
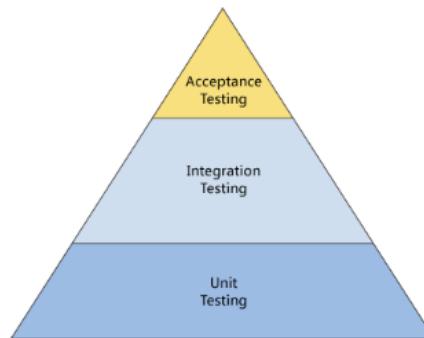
Javascript



.NET & others



What is this?



Unit Testing Techniques

Black Box Testing - Testing a system without having specific knowledge to the internal workings of the system, no access to the source code, and no knowledge of the architecture.

White/Clear Box Testing - Testing a system with full knowledge and access to all source code and architecture documents.

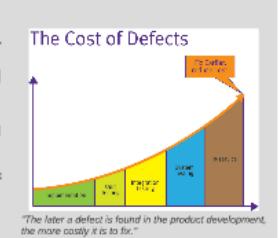
Grey Box Testing - Testing a system while having at least some knowledge of the internals of a system.

Why is this used?

- Tests one specific behavior
- Improves efficiency and reliability
- Can act as documentation of the system - the unit test itself is used to verify the design
- You can be more confident about your code
- Each test is autonomous

Benefits

- Improves efficiency and reliability
- Reduces defects in newly developed features or reduces bugs when changing the existing functionality
- Reduces cost of testing - defects are detected in an early phase
- Improves design and allows better refactoring of code
- Integration testing is easier

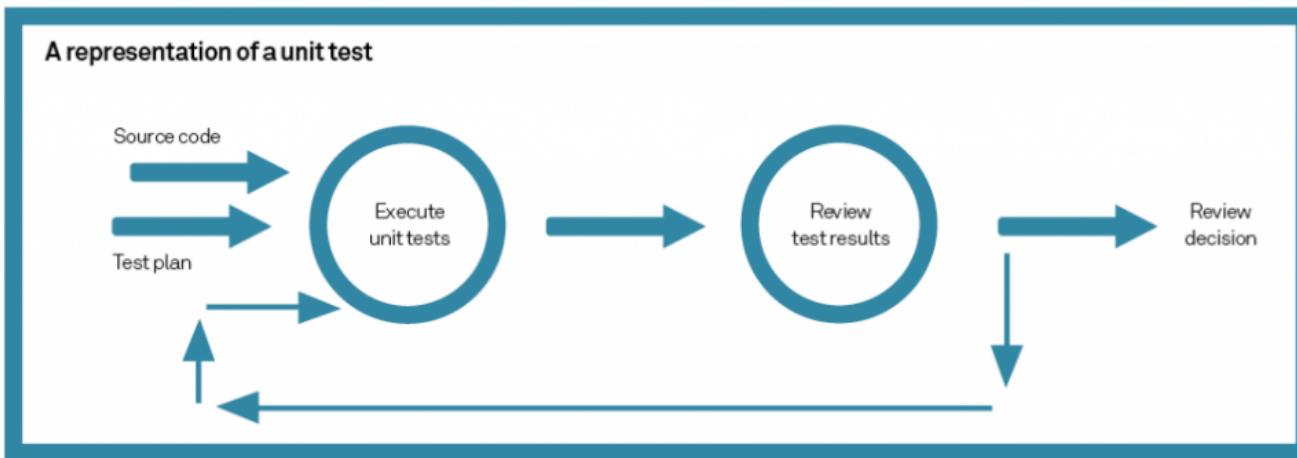
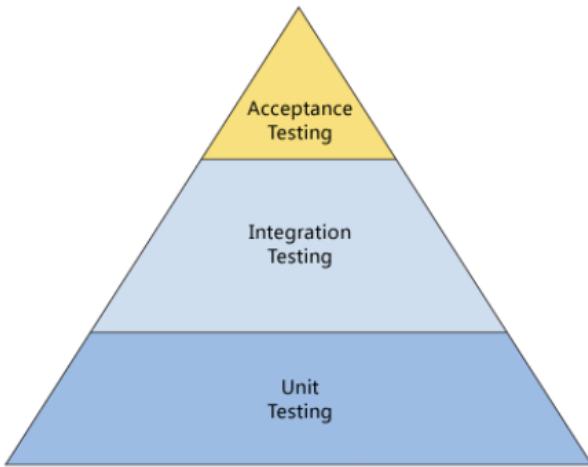


Limitations

- Time consuming
- Does not show absence of errors
- Integration errors or system errors can be missed
- Test code can have more bugs than the code it is testing
- Value and accuracy of unit tests can be diminished if initial conditions are not set correctly - there are no tests for the cases you don't have
- Does not help to better understand a requirement - can lead to a unit test checking for the wrong thing, and wrong code that passes the test



What is this?



Unit Testing Techniques

Black Box Testing - Testing a system without having specific knowledge to the internal workings of the system, no access to the

Why

- Tests
- Improv
- Can
- itself
- You
- Each

Bene

- Improve
- Reduces
- reduces
- function
- Reduces
- early ph
- Improve
- code
- Integrati

Limi



Unit Testing Techniques

Black Box Testing - Testing a system without having specific knowledge to the internal workings of the system, no access to the source code, and no knowledge of the architecture.

White/Clear Box Testing - Testing a system with full knowledge and access to all source code and architecture documents.

Grey Box Testing - Testing a system while having at least some knowledge of the internals of a system.

Why is this used?

- Tests one specific behavior
- Improves efficiency and reliability
- Can act as documentation of the system - the unit test itself is used to verify the design
- You can be more confident about your code
- Each test is autonomous

Benefits

Improves efficiency and reliability

Reduces defects in newly developed features or

The Cost of Defects

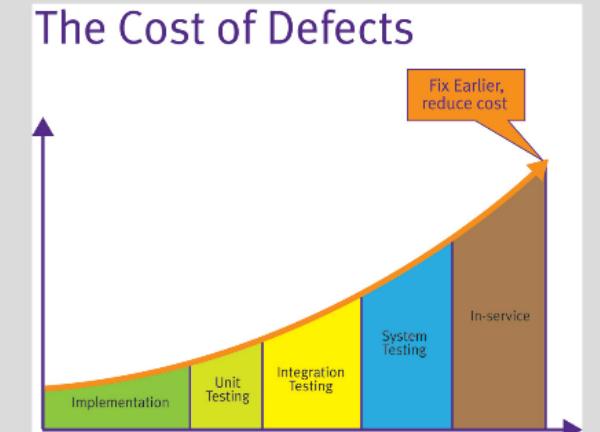
Fix Earlier,



- Each test is autonomous

Benefits

- Improves efficiency and reliability
- Reduces defects in newly developed features or reduces bugs when changing the existing functionality
- Reduces cost of testing - defects are detected in an early phase
- Improves design and allows better refactoring of code
- Integration testing is easier



"The later a defect is found in the product development, the more costly it is to fix."

Limitations



Time consuming

- Does not show absence of errors

- Integration testing is easier

"The later a defect is found in the product development, the more costly it is to fix."

Limitations

- Time consuming
- Does not show absence of errors
- Integration errors or system errors can be missed
- Test code can have more bugs than the code it is testing
- Value and accuracy of unit tests can be diminished if initial conditions are not set correctly - there are no tests for the cases you don't have
- Does not help to better understand a requirement - can lead to a unit test checking for the wrong thing, and wrong code that passes the test

Unit Testing Tools

C/C++



PHP



Javascript



.NET & others

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Web;
using System.Web.Mvc;
using System.Linq;
using HelloWorld.Controllers;
using Microsoft.VisualStudio.TestTools.UnitTesting;
```

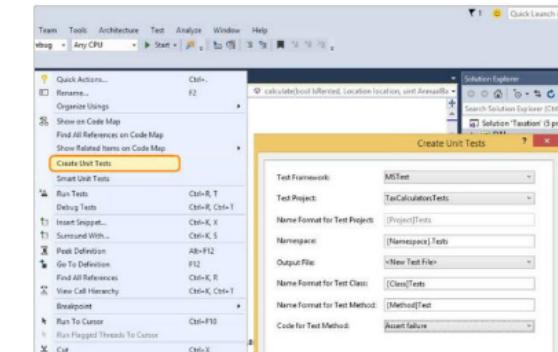
```
namespace HelloWorldTests
{
    [TestClass]
    public class HomeControllerIndexTests
    {
        [TestMethod]
        public void HomeIndexTests()
        {
            // Arrange
            HomeController controller = new HomeController();

            // Act
            ViewResult result = controller.Index() as ViewResult;

            // Assert
            Assert.AreEqual("Hello, World!", result.ViewData.Message);
        }
    }
}
```



You can also create unit test method stubs with the **Create Unit Tests** command. To learn how, see Create unit test method stubs with the **Create Unit Tests** command.

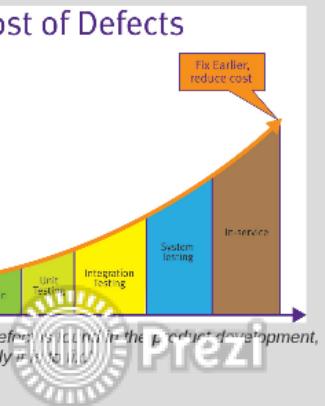


Code Coverage Tools for Javascript

- **JSCover** - free and coverage measured via a browser
- **BLANKET.JS** - deprecated for jasmine version
- **jasmine istanbul** - embedded in KARMA
- **KARMA** - spawns a web server that executes source code against test code for each of the browsers connected

Unit Testing Tools

the unit test



C/C++



PHP



Code Coverage Tools for Java

pt

ine
RMA



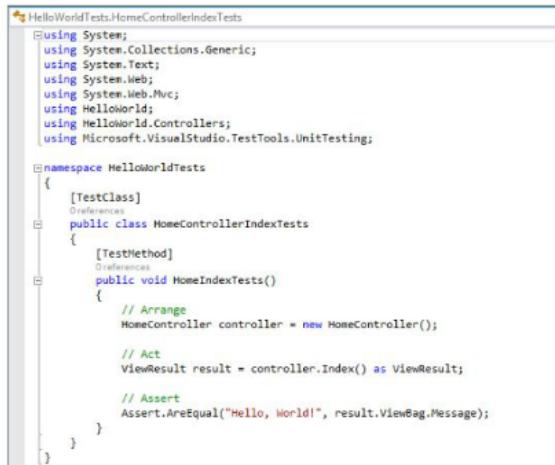
rowser

source

s



.NET & others



```
using System;
using System.Collections.Generic;
using System.Text;
using System.Web;
using System.Web.Mvc;
using HelloWorld;
using HelloWorld.Controllers;
using Microsoft.VisualStudio.TestTools.UnitTesting;

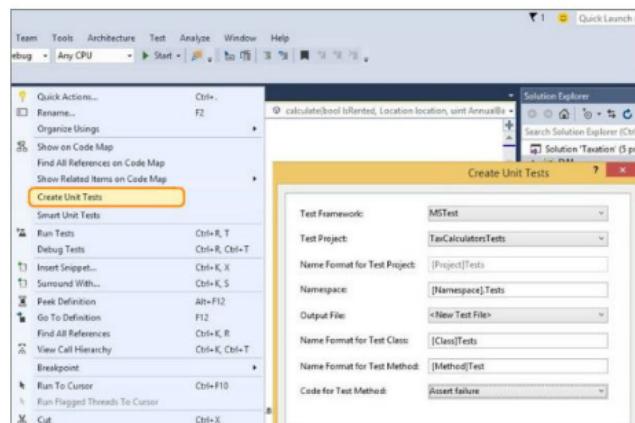
namespace HelloWorldTests
{
    [TestClass]
    public class HomeControllerIndexTests
    {
        [TestMethod]
        public void HomeIndexTests()
        {
            // Arrange
            HomeController controller = new HomeController();

            // Act
            ViewResult result = controller.Index() as ViewResult;

            // Assert
            Assert.AreEqual("Hello, World!", result.ViewBag.Message);
        }
    }
}
```



You can also create unit test method stubs with the **Create Unit Tests** command. To learn how, see [Create unit test method stubs with the Create Unit Tests command](#).



IIIIg I UUUIJ

Javascript

.NET



```
using System;
using System.Collections.Generic;
using System.Text;
using System.Web;
using System.Web.Mvc;
using HelloWorld;
using HelloWorld.Controllers;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace HelloWorldTests
{
    [TestClass]
    public class HomeControllerIndexTests
    {
        [TestMethod]
        public void HomeIndexTest()
        {
            // Arrange
            HomeController controller = new HomeController();

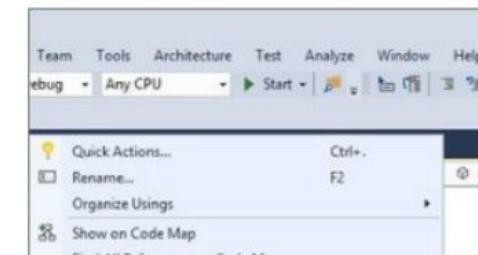
            // Act
            ViewResult result = controller.Index() as ViewResult;

            // Assert
            Assert.AreEqual("Hello World", result.ViewData["Message"]);
        }
    }
}
```

You can also create unit test method stubs with the [Create Unit Tests](#) command.



Tools for Javascript

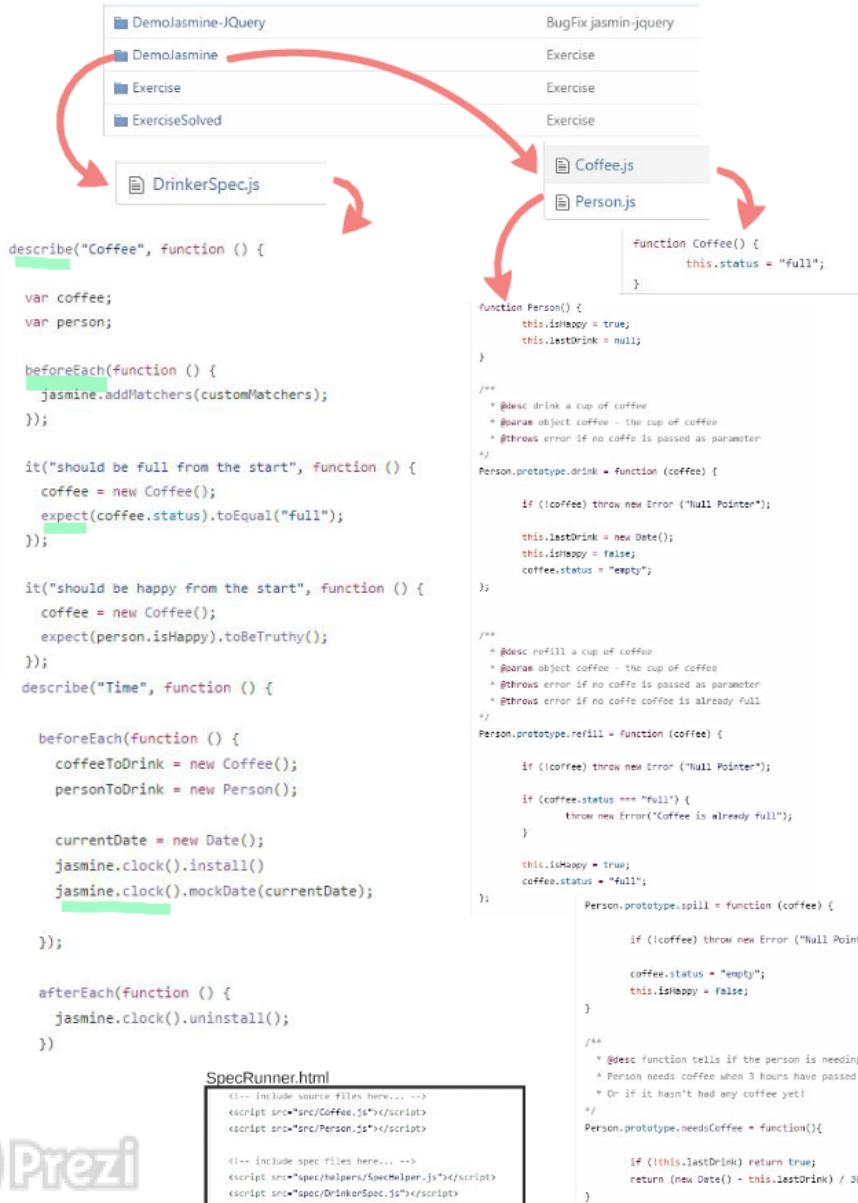
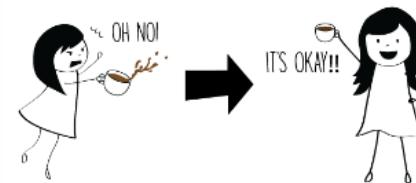


Code Coverage Tools for Javascript

- **JSCover** - free and coverage measured via a browser
- **BLANKET.JS** - deprecated for jasmine version
- **jasmine istanbul** - embedded in KARMA
- **KARMA** - spawns a web server that executes source code against test code for each of the browsers connected

Using Jasmine & KARMA

Humans are biologically built to spill coffee



Concepts:

- Specs** - function/test that contains 1 or + expectations
- Expectation** - assertion that something is T/F
- String** - title of the spec

Jasmine Setup and Teardown:

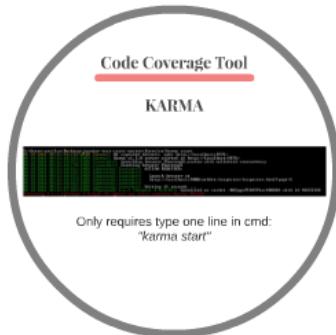
- beforeEach** - function called once before each spec
- afterEach** - function called once after each spec
- describe** - which function is called

Mocks & stubs

jasmine.clock()
 .tick(*n* milliseconds); .install(); .uninstall() .mockDate();

Nesting - Calls to **describe** can be nested, with specs defined at any level - tree of functions composition.

Before a spec is executed, Jasmine walks down the tree executing each **beforeEach** function in order.



JARMA



Concepts:

- **Specs** - function/test that contains 1 or + expectations
- **Expectation** - assertion that something is T/F
- **String** - title of the spec

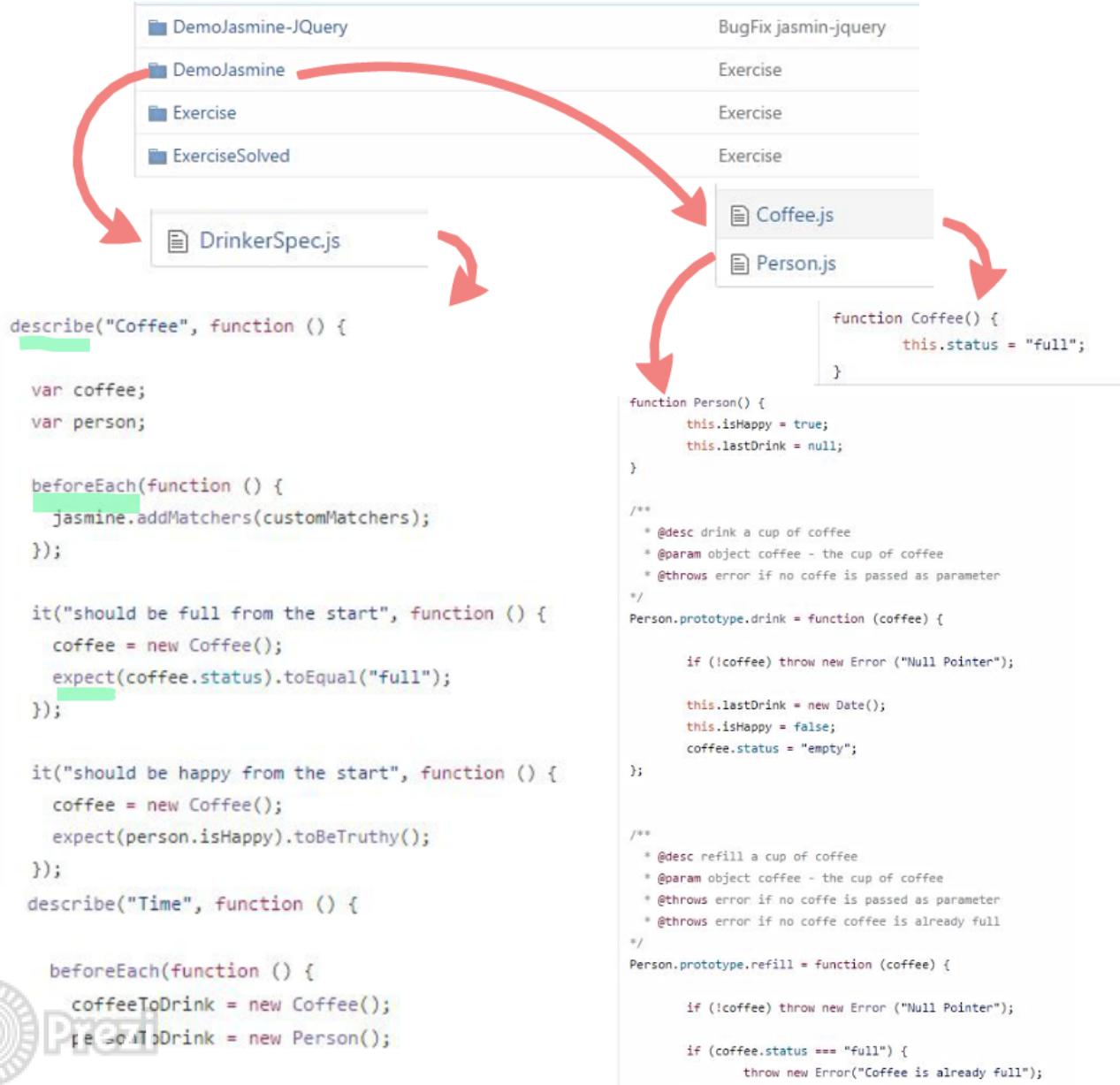
Jasmine Setup and Teardown:

- **beforeEach** - function called once before each spec
- **afterEach** - function called once after each spec
- **describe** - which function is called

Mocks & stubs

 `.tick(nºmilliseconds);` `jasmine.clock()`
`.install();` `.uninstall()` `.mockDate();`

Using Jasmine & KARMA



Concepts:

- **Specs** - function/test that contains logic
- **Expectation** - assertion that checks the result
- **String** - title of the spec

Jasmine Setup and Teardown

- **beforeEach** - function called before each spec
- **afterEach** - function called after each spec
- **describe** - which function is run

Mocks & stubs

jasmine.
.tick(*n^o milliseconds*); .install();

Nesting - Calls to **describe** can be nested at any level - tree of functions can be created. Before a spec is executed, Jasmine executes each **beforeEach** function.

- **Expectation** - assertion that something is T/F
- **String** - title of the spec

Jasmine Setup and Teardown:

- **beforeEach** - function called once before each spec
- **afterEach** - function called once after each spec
- **describe** - which function is called

Mocks & stubs

```
jasmine.clock()  
.tick(nºmilliseconds);    .install();    .uninstall()    .mockDate();
```

Nesting - Calls to **describe** can be nested, with specs defined at any level - tree of functions composition.

Before a spec is executed, Jasmine walks down the tree executing each **beforeEach** function in order.

```

it("should be full from the start", function () {
  coffee = new Coffee();
  expect(coffee.status).toEqual("full");
});

it("should be happy from the start", function () {
  coffee = new Coffee();
  expect(person.isHappy).toBeTruthy();
});

describe("Time", function () {

beforeEach(function () {
  coffeeToDrink = new Coffee();
  personToDrink = new Person();

  currentDate = new Date();
  jasmine.clock().install()
  jasmine.clock().mockDate(currentDate);
})

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

```

SpecRunner.html

```

<!-- include source files here... -->
<script src="src/Coffee.js"></script>
<script src="src/Person.js"></script>

<!-- include spec files here... -->
<script src="spec/helpers/SpecHelper.js"></script>
<script src="spec/DrinkerSpec.js"></script>

```

```

Person.prototype.drink = function (coffee) {
  if (!coffee) throw new Error ("Null Pointer");

  this.lastDrink = new Date();
  this.isHappy = false;
  coffee.status = "empty";
};

/**
 * @desc refill a cup of coffee
 * @param object coffee - the cup of coffee
 * @throws error if no coffee is passed as parameter
 * @throws error if no coffee coffee is already full
 */
Person.prototype.refill = function (coffee) {
  if (!coffee) throw new Error ("Null Pointer");

  if (coffee.status === "full") {
    throw new Error("Coffee is already full");
  }

  this.isHappy = true;
  coffee.status = "full";
};

Person.prototype.spill = function (coffee) {
  if (!coffee) throw new Error ("Null Pointer");

  coffee.status = "empty";
  this.isHappy = false;
};

/**
 * @desc function tells if the person is needing coffee
 * Person needs coffee when 3 hours have passed since last drink,
 * Or if it hasn't had any coffee yet!
 */
Person.prototype.needsCoffee = function(){
  if (!this.lastDrink) return true;
  return (new Date() - this.lastDrink) / 3600000.0 >= 3.0;
}

```

.tick(n^o milliseconds)

Nesting - Calls to `it` at any level - tree of specs
Before a spec is executing each `before`

Jasmine

- Used for manipulation, handling and animation
- Works across several browsers.
- Allows layout validation of web pages.

Jasmine Plugin

jQuery

- Used for HTML manipulation, event handling and animation.
- Works across several browsers.
- **Allows layout validation of web pages.**

```
describe("JQuery", function () {  
  
    var fixture;  
    beforeEach(function () {  
        loadFixtures('test.html');  
    })  
  
    it('should have a cool title', function () {  
  
        title = $('h1')  
        expect(title).toHaveText("MESW Login");  
    });  
});
```



Username

Password

Submit

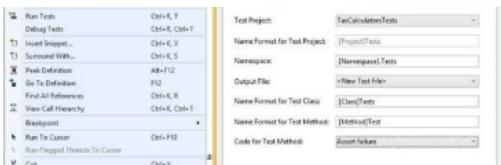
Code Coverage Tool

KARMA

```
C:\Users\arsilva\Desktop\jasmine-test-cases-master\Exercise>karma start
04 10 2016 09:21:10.541:WARN [karma]: No captured browser, open http://localhost:9876/
04 10 2016 09:21:10.552:INFO [karma]: Karma v1.3.0 server started at http://localhost:9876/
04 10 2016 09:21:10.553:INFO [launcher]: Launching browser PhantomJS_custom with unlimited concurrency
04 10 2016 09:21:10.570:INFO [launcher]: Starting browser PhantomJS
04 10 2016 09:21:10.651:INFO [phantomjs.launcher]: ACTION REQUIRED:
04 10 2016 09:21:10.652:INFO [phantomjs.launcher]:
04 10 2016 09:21:10.653:INFO [phantomjs.launcher]: Launch browser at
04 10 2016 09:21:10.654:INFO [phantomjs.launcher]:   http://localhost:9000/webkit/inspector/inspector.html?page=2
04 10 2016 09:21:10.655:INFO [phantomjs.launcher]: Waiting 15 seconds ...
04 10 2016 09:21:27.283:INFO [PhantomJS 2.1.1 (Windows 8 0.0.0)]: Connected on socket /#U2pgvF5h9YFloc4DAAAA with id 46832368
PhantomJS 2.1.1 (Windows 8 0.0.0) TO should not allow nameless Movies FILED
```

Only requires type one line in cmd:
"karma start"

- **JSCover** - free and coverage measured via a browser
- **BLANKET.JS** - deprecated for jasmine version
- **jasmine istanbul** - embedded in KARMA
- **KARMA** - spawns a web server that executes source code against test code for each of the browsers connected



Exercise



Can you find the errors in source code?
Achieve 100% coverage and add new test case(s).

[all files](#) [src/](#)

70% Statements 21/30 42.86% Branches 6/14 77.78% Functions ??/6 74.07% Lines 26/27

File	Statements	Branches	Functions	Lines
Movie.js	100%	6/6	100%	3/3
Television.js	62.5%	15/24	42.86%	4/6

Follow the instructions provided in:

<https://github.com/mzamith/jasmine-test-cases>



References

- iTGROW Software and Systems Training
- ISTQB Foundation Level (materials provided during the course)
- Software testing & QA blog by TestFort, 2016
- "MSDN Developer Network", available at: <https://msdn.microsoft.com/en-us>
- "Saujana Magazine"
- "Tutorialspoint", available at: <https://www.tutorialspoint.com>
- "Embedded Insights", available at: <http://www.embeddedinsights.com/>

“ Fools ignore complexity. Pragmatists suffer it.
Some can avoid it. Geniuses remove it. ”

Alan J. Perlis

References

- iTGROW Software and Systems Training
- ISTQB Foundation Level (materials provided during the course)
- Software testing & QA blog by TestFort, 2016
- "Microsoft Developer Network", available at: <https://msdn.microsoft.com/en-us>
- Gadget Magazine
- "Tutorialspoint", available at: <https://www.tutorialspoint.com>
- "Embedded Insights", available at: <http://www.embeddedinsights.com/channels/topics/unit-test/>
- "Jasmine" GitHub, available at: <http://jasmine.github.io/2.0/introduction.html>
- "KARMA", available at: <https://karma-runner.github.io/0.8/intro/configuration.html>

**Fools ignore complexity. Pragmatists suffer it.
Some can avoid it. Geniuses remove it.**

Alan J. Perlis



UNIT TESTING

Ana Rita Silva, Manuel Zamith

TVVS

Friday, October 7, 2016

What is this?

Unit Testing Techniques

- Black Box Testing:** Testing a system without having specific knowledge to the internal technology of the system, no actions to the source code, and no knowledge of the variables.
- White/Clear Box Testing:** Testing a system with full knowledge and access to all source code and architecture documents.
- Grey Box Testing:** Testing a system while having at least some knowledge of the internals of a system.

Why is this used?

- Tests have specific behavior
- Improves efficiency and reliability
- Can verify the whole system - the unit test itself is used to verify the design
- You can be more confident about your code
- Each test is autonomous

Benefits

- Improves efficiency and reliability
- Tested areas in many thousands times, so the whole system is tested
- Knowledge of testing - which are included in it: requirements, design and actions, better refactoring, and integration testing is easier

Limitations

- Time consuming
- Only tests what is explicitly coded
- Intelligence of system being tested can be tested
- Value and usefulness of Unit tests can be often debated - what and when to write them, how much time to spend on them, etc.
- Does not help to detect architectural requirements - can lead to a false sense of security for the system being tested

Unit Testing Tools

C/C++

- VECTOR
- CUTE++
- CPPUTEST

Javascript

- Jasmine
- Karma
- Chai
- mocha
- QUnit

.NET & others

- Microsoft Test Manager
- Visual Studio Test
- NUnit
- SpecFlow
- SpecFlow+Runner

Code Coverage Tools for Javascript

- JSCover - free and coverage measured via a browser
- BLANKETJS - deprecated for jasmine version
- jasmine-istanbul - embedded in KARMA
- KARMA - spawns a web server that executes source code against test code for each of the browsers connected

Using Jasmine & KARMA

Codebase

- Specs:** Functional test that contains 1 or + expectations
- Expectations:** Assertion that something IS
- String:** like the spec

Jasmine Specs and TestRunners

- Specs:** A file containing one or more specs. It must contain at least one describe block. It must be placed before each spec.
- afterEach:** Function called once after each spec.
- describe:** Function it called

Mode & setup

```
jasmine.getEnv().createTestEnv().init();
```

Karma

```
Gruntfile.js: karma.conf.js: mocha, sinon, sinonchai, jest, QUnit
```

Notes: Code to describe can be injected, with specs defined at any level - type of function connection. Before a spec is executed, Jasmine walks down the tree, executing each beforeEach function in order.

Exercise

Can you find the errors in source code?
Achieve 100% coverage and add new test case(s).

File: test.html

Pass: 0% | Pending: 100% | Fail: 0% | Error: 0% | Skipped: 0% | Total: 100%

Please follow the instructions provided in:
http://www.karma-runner.org/guide/test_writing.html

References

- “The Art of Agile Development”
- “Continuous Integration: Practical Techniques for Integrating People, Code, and Tools”
- “Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation”
- “Continuous Deployment: Automating the Software Delivery Pipeline”
- “Continuous Deployment: Continuous Delivery and Deployment”
- “Continuous Deployment: Continuous Delivery and Deployment”

“Fools figure complexity. Pragmatists suffer it. Some can avoid it. Geniuses remove it.”
Alan J. Perlis

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO