Introduction

- Allen Conway

  - **Blog**: http://www.AllenConway.net
  - **Twitter**: @AllenConway, http://www.twitter.com/AllenConway
  - **GitHub**: https://github.com/AllenConway
  - **Email**: AllenC@Magenic.com

What we will cover

- Unit Testing Foundation
- JavaScript Unit Testing Frameworks
- Test Runners
- Writing JS Unit Tests using Jasmine
- Mocking in unit tests
- Unit Testing TypeScript
- Unit Testing Angular

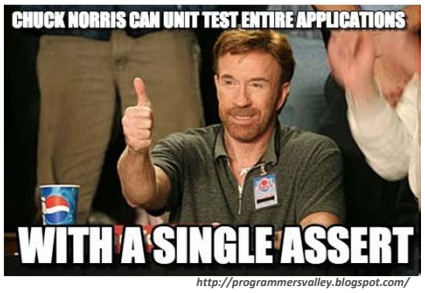
Laying the Foundation

- What is Unit Testing?
- Do I need to unit test my applications?

Why do you need to write tests?

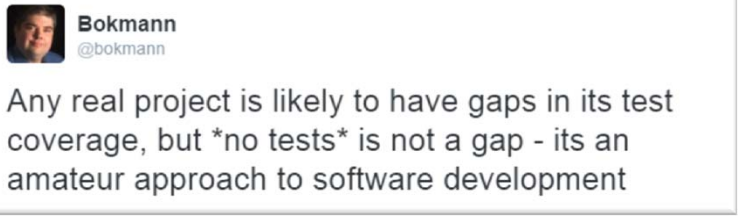
“Because you're not Chuck Norris. Chuck Norris' code tests itself, and it always passes, with 0ms execution time.”

http://andyshora.com/unit-testing-best-practices-angularjs.html



http://programmersvalley.blogspot.com/

Some Perspectives on Unit Testing



Bokmann
@bokmann

Any real project is likely to have gaps in its test coverage, but *no tests* is not a gap - its an amateur approach to software development

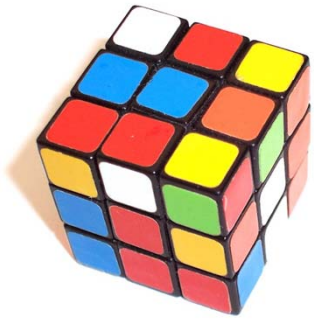
Tim Ottinger
@tottinge

Not good messaging: "Don't _waste_ dev time writing tests, instead _spend_ your evenings, weekends, and holidays fixing buggy code"

Void of Unit Tests?

- What happens if we don't unit test?
- How do we *truly* verify code works as expected?
- Physical Integration Testing and QA
- The domino effect on code changes

Challenges of Unit Testing



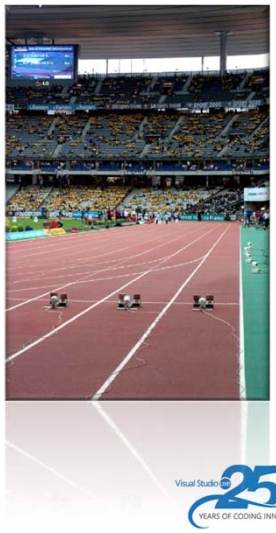
- More code
- New paradigm
- Existing implementations
- Knowing *what* and *how* to test
- The art of creating *good and meaningful unit tests*

Writing Testable Code

- SoC
- Dependency Injection and IoC
- Using abstractions and isolating code
- Orthogonal design
- Large intermingled methods and tightly coupled code != easier unit tests

Moving Forward

- 1..n tests > 0 tests
- Write testable code
- Minimize Distractions



JS Unit Testing Frameworks

- Allows writing unit tests for JavaScript
- Popular JS Testing Frameworks
  - **Jasmine**
  - Mocha
  - Protractor (E2E)
  - Jest (React)

JavaScript Test Runners

- Allows running JS unit tests
- Provides output and results
- Can typically be integrated into a CI build
- Popular JS Test Runners
  - **Jasmine**
  - Karma (Angular)
  - Jest (React)
  - Ava
  - QUnit
  - Protractor (E2E testing)
  - Chutzpah (VS)
  - ReSharper
- Headless Browser
  - Chrome
  - PhantomJS*

The Jasmine Framework

- BDD framework for testing JavaScript code
  - Expressive from an end user's point of view
- Minimal Learning Curve
- Download/Install
  - NPM
  - Yarn
  - Standalone .zip
  - NuGet*
- Latest Version 3.2.1

Writing Jasmine JavaScript Unit Tests

- Organize 'Tests' in symmetric structure
- Test runner files dictate dependencies and run tests
- Specs contain the physical tests
- describe()
  - it()
    - beforeEach()
    - afterEach()
    - Matchers (expectations)
- Tests read like acceptance criteria of a user story
  - describe + it blocks
- BDD tests can read as GWT
  - Can also organize in tests as AAA

Jasmine Matchers

- In Jasmine, expectations are chained with 'Matchers'
- Several built in matchers for testing
  - `expect(actual).toBe(expected);`
  - `expect(actual).toEqual(expected);`
  - `expect(actual).toBeNull();`
  - `expect(actual).toBeTruthy();`
- Add **not** before matcher for negative assertion
  - `expect(actual).not.toContain(expected);`
- Ability to create custom matchers
  - Full list
    - https://github.com/jasmine/jasmine/tree/master/src/core/matchers

Jasmine Test Runner

- Simply open spec runner file in a browser
- Easy to read HTML output
- Filtering functionality
- Ability to run a single test
- Good for running a few tests

Unit Testing Async Code

- 3 methods for async testing
  - Callbacks
    - Use the `done` function
  - Promises
    - Support added in Jasmine 2.7
    - Return `Promise` and `resolve()`
  - `async`
    - Support added in Jasmine 2.7
- Mock clock
  - `jasmine.clock().tick` fucntion
- Recommend using a mock
  - ajax, $http/XHR calls

Jasmine Spies

- Used for mocking
  - Isolate code from dependencies
- Focuses on 'unit' of code as opposed to integration testing with dependencies
  - Removes dependencies on:
    - Network connections
    - Services
    - External files
    - External objects
    - Database calls
    - 3rd party libraries
- Allows asserting method behavior

Creating Jasmine Spies

- Spy creates a proxy object
- spyOn()
  - Used for existing functions
  - Spies on an object's method
- createSpy()
  - Creates a spy with no implementation behind it
- createSpyObj()
  - Creates a mock with multiple spies
  - Takes an array of strings

Matchers in Jasmine Spy

- Primary spy matchers
  - `toHaveBeenCalled();`
    - Returns true if the spy was called
  - `toHaveBeenCalledWith();`
    - Returns true if argument list matches calls to the spy
- Preface with `not` to negate the matcher

Unit Testing TypeScript

- Write unit tests using TypeScript
- TypeScript is a superset of JavaScript
  - Any JavaScript is valid TypeScript
- Compile .ts files
  - Target .js output in test runner or configuration
- Add Type Definition Files
  - jasmine.TypeScript.DefinitelyTyped

Using a Mocking Framework

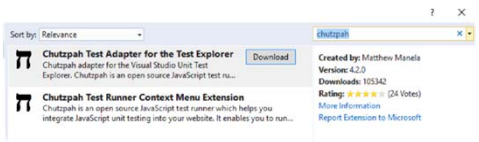
- Streamlines unit test work
- JS/TS Mocking Frameworks
  - TypeMoq
    - Analogous to frameworks like ‘moq’ in C#
  - ts-mockito
    - Inspired by mockito
  - SinonJS
    - Standalone spies, stubs, mocks

Visual Studio Test Runners

- Allows running unit tests directly *inside* VS
- No need to jump out to command line or browser
- Few choices but they can work (ES5)
  - Rely on using PhantomJS Headless Browser
    - **Chutzpah**
    - **ReSharper**
- Consider VS Code

Using Chutzpah in Visual Studio

- Direct Visual Studio Integration for VS12+
  - NuGet: Install-Package Chutzpah
- Supports: **Jasmine**, **Mocha**, & **QUnit**
- Right-click menu context options
  - Run the individual spec
  - Run the chutzpah.json file



Using ReSharper in Visual Studio

- Popular productivity tool used in Visual Studio with test runner capabilities
- Not required for running Jasmine Tests
  - Alternative test runner choice for VS.NET
- Uses default browser for testing
  - Can also be configured to use PhantomJS

Demo App - Auto Performance Angular App

Unit Testing Angular

- Nature of Angular aligns well for unit testing
  - Best practices still advised
- Test services/providers in isolation
- Test Components using the Angular Test Bed
- Somewhat boiler plate 'beforeEach' setup code
  - Both Angular and AngularJS

Mocking in Angular

- DI built into Angular
  - Testbed assists DI with service creation
- Testing dependencies
  - Angular
    - @angular/core/testing
  - AngularJS
    - ngMock via angular-mocks.js

Useful References

- Jasmine Documentation
  - https://jasmine.github.io/pages/docs_home.html
- Unit Testing Angular
  - https://angular.io/guide/testing
  - https://docs.angularjs.org/guide/unit-testing (v1.x)
- Chutzpah GitHub Documentation
  - https://github.com/mmanela/chutzpah
- GitHub Samples
  - https://github.com/AllenConway/

Thank you! Q&A

http://c1.staticflickr.com/1/28/65098350_b7bd96f38_b.jpg

Allen Conway

- **Blog**: http://www.AllenConway.net
- **Twitter**: @AllenConway
- **GitHub**: https://github.com/AllenConway
- **Email**: AllenC@Magenic.com

Thank you for attending Visual Studio LIVE!