# Unit Testing in AngularJS

Alex Wong

## Section A: Unit Testing Overview

http://notafraidofwong.blogspot.com/search/label/Unit%20Testing

## Getting Started

- http://notafraidofwong.blogspot.com/2018/07/unit-testing.html

## Section B: Unit Testing Deep Dive

## How to actually Unit Test? Stubs, Mocks.etc

- http://notafraidofwong.blogspot.com/2018/07/unit-testing-test-doubles-stubs-mocksetc.html

## Section C: AngularJS Unit Testing Framework - Karma

## Karma

- Quoting from Karma website (https://karma-runner.github.io/latest/index.html)
  - *“On the AngularJS team, we rely on testing and we always seek better tools to make our life easier. That's why we created Karma - a test runner that fits all our needs.”*

- A tool which spawns a web server for users to test their JavaScript source code.
- More info at https://karma-runner.github.io/3.0/intro/how-it-works.html

## How Karma works?

1. Starts local web server and starts one or more browsers.
2. Karma instructs client (with hosted js via web server) to execute tests.
3. The result of the test is reported back to the server via websockets.
4. Reporters listen on ws messages and process it
    - It may print, save to file or forward data to another server.
    - Main karma reporter is **Jasmine**.

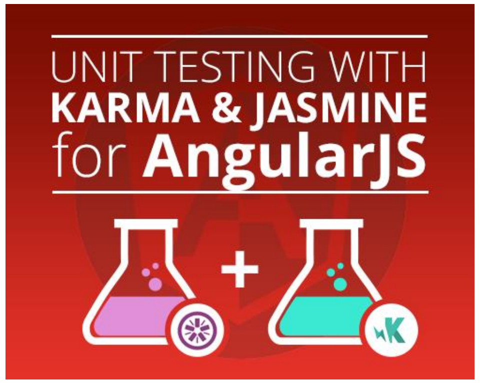
- Note that Karma has many variations and options that may cause different workflow with different configurations.
- More detailed steps are in https://karma-runner.github.io/3.0/intro/how-it-works.html

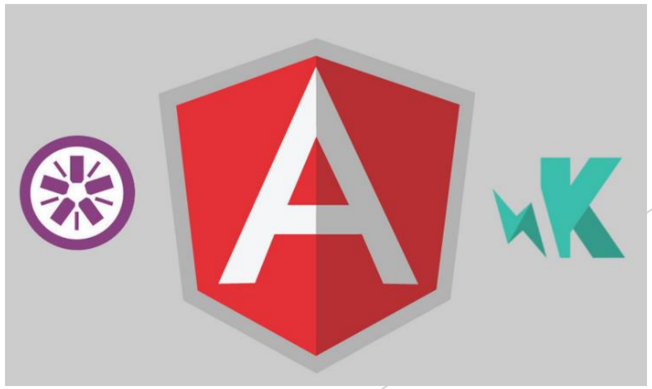
## What is Jasmine?

- Jasmine is a **Behavior Driven Development testing framework** for JavaScript.
- It does not rely on browsers, DOM, or any JavaScript framework. Thus it's suited for websites, Node.js projects, or anywhere that JavaScript can run.
- More info in [https://github.com/jasmine/jasmine](https://github.com/jasmine/jasmine)

# Karma vs Jasmine

- If you only had Jasmine (or Mocha), you are running all your code with the Node virtual Machine.
- Adding Karma to the mix will run your test suite with the engine of other browsers.
- By doing this, you get are testing more closely with the browser environment. It will also be easier to test DOM related code.
- Source: [https://stackoverflow.com/questions/26032124/karma-vs-testing-framework-jasmine-mocha-qunit](https://stackoverflow.com/questions/26032124/karma-vs-testing-framework-jasmine-mocha-qunit)



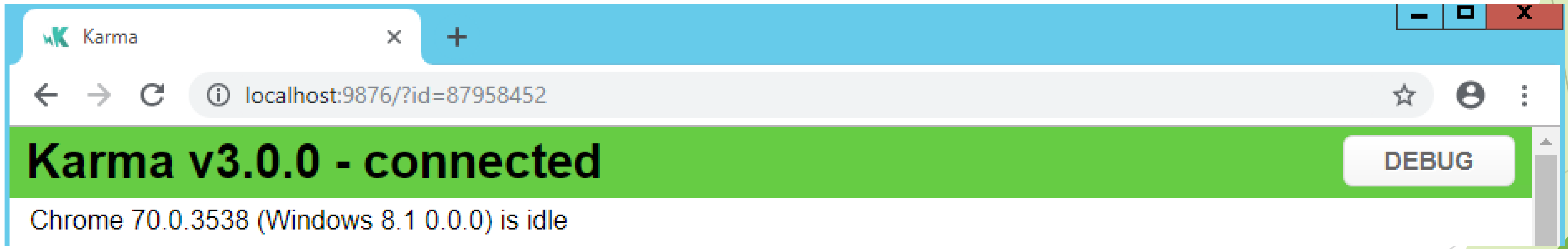


## Executing Karma Unit Tests

- You must have karma-cli installed.
  - You can do that via “npm install –g karma-cli”
- You need to have some karma unit testing source code
  - I already wrote this. To find this: ~/webui/test/unit

# Executing Karma Unit Tests

- To run this:
  - Open unit test source code in terminal.
  - Run ‘npm install’
  - Run ‘karma start’
    - If your karma’s browser setting is set to ‘Chrome’, you will see this pop up.



Karma v3.0.0 - connected

Chrome 70.0.3538 (Windows 8.1 0.0.0) is idle

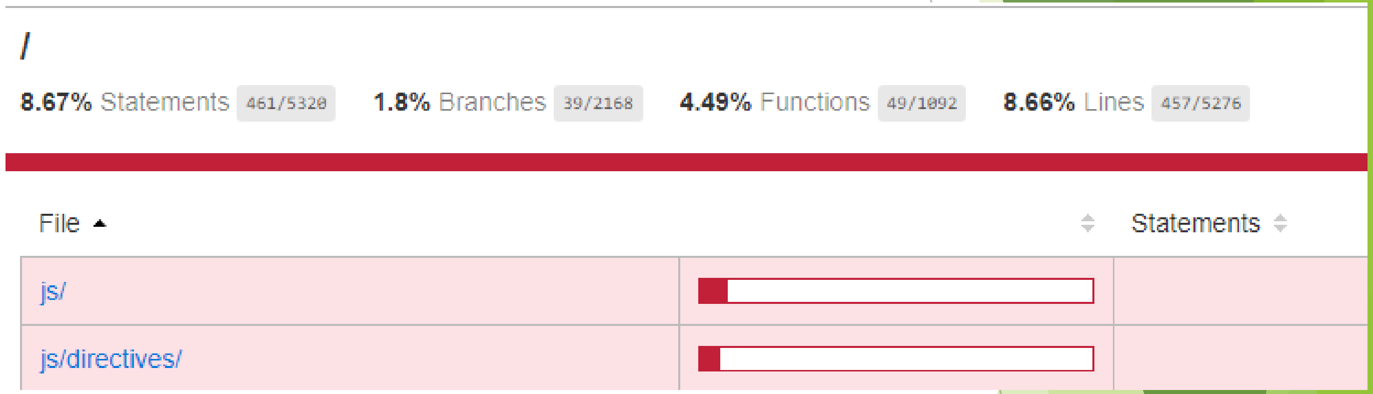
## Karma plugins: Karma-coverage

- A karma plugin. Generate code coverage.
- Behind the scenes, it uses **Istanbul**, a comprehensive JavaScript code coverage tool.
- In karma.conf.js, these are the important configurations:

  -
    ```
    reporters: ['progress', 'coverage'],
    coverageReporter: {
        type : 'html',
        dir : 'test/unit/coverage/' // output destination folder for coverage report.
    },
    ```
- More info: https://github.com/karma-runner/karma-coverage

# Karma plugins: Karma-coverage

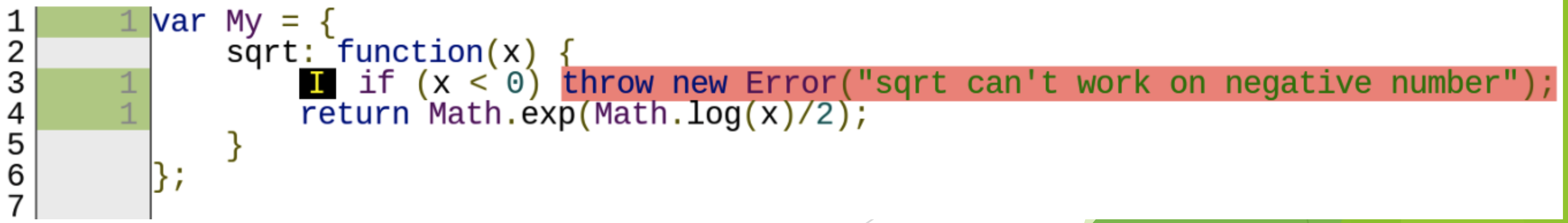
- Example:
- First screenshot displays code coverage **by folder**.
- Second screenshot displays code coverage **per script**. (Line coverage)
  - Red is untouched. Not red
- More info: [https://github.com/karma-runner/karma-coverage](https://github.com/karma-runner/karma-coverage)



/

8.67% Statements 461/5320 1.8% Branches 39/2168 4.49% Functions 49/1092 8.66% Lines 457/5276

| File ▲ | ⬍ | Statements ⬍ |
|---|---|---|
| js/ | 🟥 | |
| js/directives/ | 🟥 | |

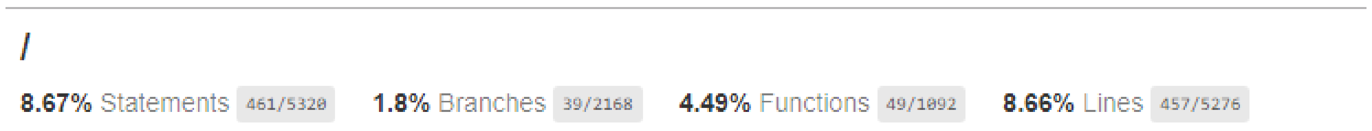


```
1 var My = {
    sqrt: function(x) {
1       if (x < 0) throw new Error("sqrt can't work on negative number");
1       return Math.exp(Math.log(x)/2);
    }
};
```

# Coverage

- There are several types of coverages; in karma-coverage, we have 4.

1. Line Coverage
2. Statement Coverage
3. Branch Coverage
   - Sometimes, your code can branch into different possibilities; such as using an if-else statement.
   - Branch coverage measures how much of the branches are tested/executed
4. Function Coverage



/

8.67% Statements 461/5320 1.8% Branches 39/2168 4.49% Functions 49/1092 8.66% Lines 457/5276

## Junit Report (Karma-junit-reporter)

- A Karma plugin. Report results in junit xml format.
- Reason we need this is to integrate to Jenkins workflow. (explained in next slide)
- More info: [https://github.com/karma-runner/karma-junit-reporter](https://github.com/karma-runner/karma-junit-reporter)

# Jenkins Integration

- Jenkins CI is one of the most popular continuous integration servers.
- To integrate with Karma, karma needs to export a Junit test result let Jenkins access it.
- To Configure Karma to export Junit test result.
  - Make sure karma-junit-reporter is installed.
  - Make following changes to your karma.conf.js file:
    - singleRun: true,
      reporters: ['dots', 'junit'],
      junitReporter: { outputFile: 'test-results.xml' },
    - Should configure outputFile to your preferred destination.
- More instructions: [http://karma-runner.github.io/3.0/plus/jenkins.html](http://karma-runner.github.io/3.0/plus/jenkins.html)

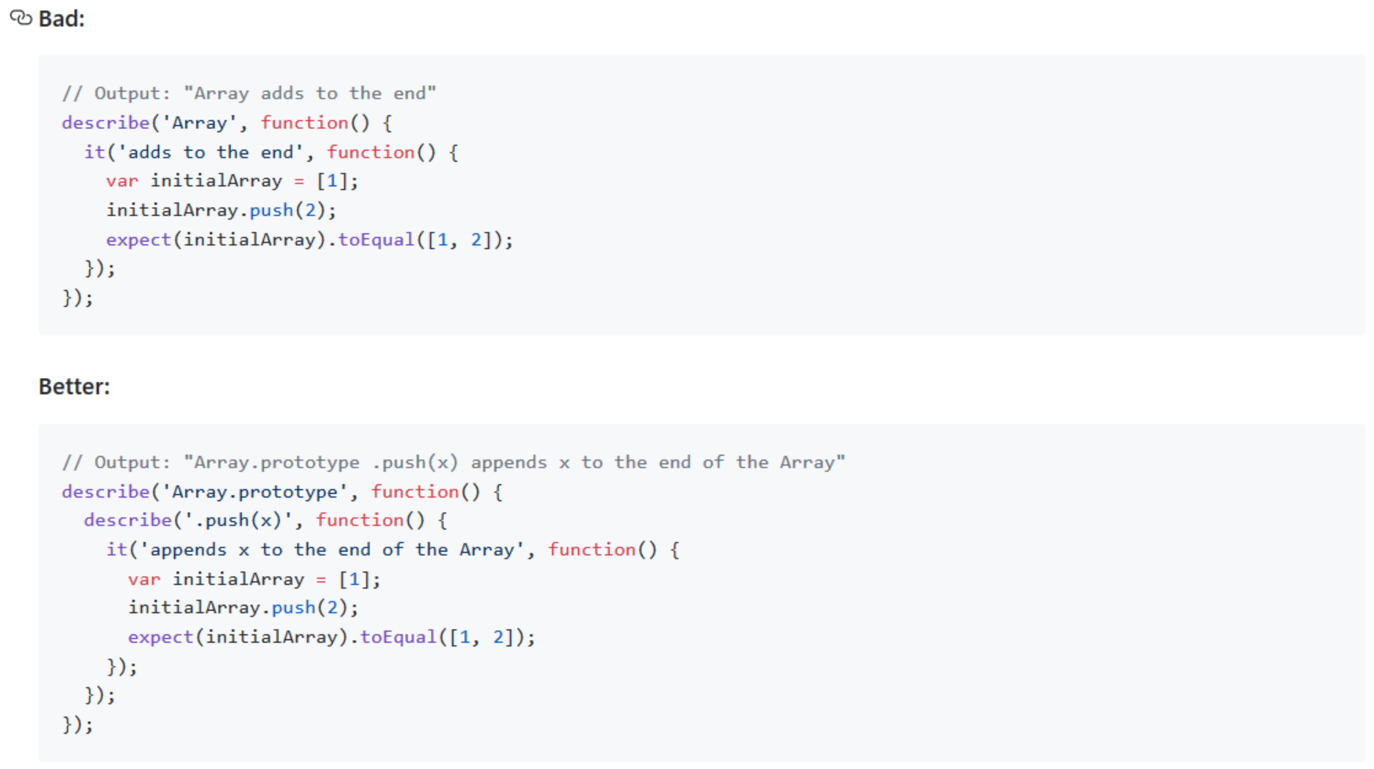
## Section D: Jasmine Specs (test cases)

# Notes on Jasmine

- Basic Jasmine walkthrough:
  - https://jasmine.github.io/tutorials/your_first_suite
- Jasmine Style Guide
  - https://github.com/CareMessagePlatform/jasmine-styleguide#speak-human
- Demo:
  - karma.conf.js walkthrough
    - Files
    - Browser – headless option
  - loadDetailsErrorController.spec.js
    - Setting up service, factory, controller
    - Talk about issues I found in my own code.
      - $scope.onUpdateErrType
    - Promises

# Notes on Jasmine

- Naming Conventions
  - Source: [https://github.com/CareMessagePlatform/jasmine-styleguide#speak-human](https://github.com/CareMessagePlatform/jasmine-styleguide#speak-human)



**Bad:**

```
// Output: "Array adds to the end"
describe('Array', function() {
  it('adds to the end', function() {
    var initialArray = [1];
    initialArray.push(2);
    expect(initialArray).toEqual([1, 2]);
  });
});
```

**Better:**

```
// Output: "Array.prototype .push(x) appends x to the end of the Array"
describe('Array.prototype', function() {
  describe('.push(x)', function() {
    it('appends x to the end of the Array', function() {
      var initialArray = [1];
      initialArray.push(2);
      expect(initialArray).toEqual([1, 2]);
    });
  });
});
```

## Section E: Best Practices

- http://notafraidofwong.blogspot.com/2018/08/best-practices-of-unit-testing.html
- http://notafraidofwong.blogspot.com/2018/08/unit-testing-unit-test-as-bug-report.html