Unit testing with Karma and Jasmine for AngularJS

A small app can have errors and it’s bad practice to have an untested app.

Besides, a small app is the perfect place to get started in using proper unit testing because there won’t be many tests and the functionality will be less complex.

**First of all, what is Unit Testing?**

Unit testing is very important for both the quality of the code and the well being of the developer. It allows for them to freely change their code and not just hope that it’s not a breaking change, as the test will recognize the break straight away. This will minimize the number of regressions in the code.

Unit tests are all about splitting your code into small testable modules with each module having its own functionality and level of abstraction. This is where AngularJS shines.

An added bonus is that the unit tests will act as a type of documentation because it describes the expected behavior of the functions and modules. Unit tests don’t care how your code does the task, it just cares about the results.

**Okay, but when should I write unit tests?**

Short answer – the start of the project is the ideal time. You will have all your tests and then can work toward writing your code with passing them in mind.

Write as many as you can possibly think of at the start, and if new components appear, then they should be tested too; a more iterative process of revisiting tests should be used.

**What testing framework should I use?**

AngularJS can be tested in a lot of ways, by lots of testing frameworks, including but certainly not limited to **mocha, jasmine, qunit, and sin** on.

After you pick your testing framework, you can now choose a task runner. The most well known include **Karma** and **Protractor** (both by the Angular team).

We’ll need to install some dependencies for running unit tests with Karma and Jasmine.  
To do this, you should run the following lines on the command line:

npm install karma --save-dev

npm install karma-phantomjs-launcher --save-dev

npm install karma-jasmine --save-dev

**How to design a test?**

Scenario

* “A calculator that adds and subtracts”

This scenario can be broken into features, which will in turn be broken into units of testable code.

The calculator has two main features, adding and subtracting numbers. These features can be split into units, such as the methods for addition and subtraction. These methods need to be tested and there may be many different cases that need to be taken into account. A possible example is below, but you may come up with some more tests.

describe('Calculator ', function() {

// necessary

it('should add two numbers correctly', function() {});

it('should subtract two numbers correctly', function() {});

// helpful but not needed

it('should add negative numbers', function() { });

it('should reject non numbers', function() { });

});

The “describe()” method defines our test suite. This takes a string with the convention that it describes the module which is being tested.

An “it()” is a proper test spec. Put what you want to be tested here, the convention is that it should read as a sentence, e.g. ‘it is expected to do this’ reads as “it is expected to do this”

**How do I use it with my AngularJS app?**

Here is an example of a very basic controller. It only has a scope variable and we will test if this value is correct.

var myApp = angular.module('myApp',[]);

myApp.controller('HelloWorldController', ['$scope', function($scope) {

$scope.greeting = 'Hello World!';

}]);

**Jasmine Test**

describe(‘Hello World example ’, function() {

beforeEach(module(‘myApp’));

var HelloWorldController,

scope;

beforeEach(inject(function ($rootScope, $controller) {

scope = $rootScope.$new();

HelloWorldController = $controller('HelloWorldController', {

$scope: scope

});

}));

it('says hello world!', function () {

expect(scope.greeting).toEqual("Hello world!”);

});

});

**Run your test with:**

karma start karma.conf.js

Every “beforeEach()” function is executed right before the tests are conducted. In the above case the function gets access to your angular module and then injects the controller and scope, so that it can be tested. This allows you to set up for each assertion, allowing the same clean set up for all your tests.

Some of the possible expectations include:

* **toEqual, toBe, toBeTruthy, toBeFalsy**
* **toBe for negation**

Jasmine has you covered if you want to ignore some of the tests temporarily. “xdescribe” will ignore a “describe” and “xit” will ignore a specific test.

https://blog.logentries.com/2015/01/unit-testing-with-karma-and-jasmine-for-angularjs/