Testing AngularJS

by Vlad Costel Ungureanu
for “Learn Stuff” and “CGM Romania”

Unit Testing

- ✓ Unit testing does not ensure correctness or acceptance
- ✓Unit testing does not ensure that the code is correct and does not ensure that the feature is actually done
- ✓Unit testing is meant to ensure that after a functionality is implemented, ulterior changes do not alter the expected behaviour of a feature
- ✓Unit testing ensures that refactoring and changes do not alter previously existing functionality
- ✓Unit testing does not justify bad code or bad practices
- ✓It is a myth that unit testing makes code quality better

Unit Testing

- ✓ As the name suggests, a unit test focuses on a single unit of code, which in case of JS is an callable object

- ✓ Unit testing is meant to test that a certain functionality provides the same output for the same input for any given numbers of executions

- ✓ Unit testing needs to be specific, controllable, repeatable and predictable

- ✓ A unit test needs to focus a single specific functionality and verify a single behaviour

(the boilerplate code for some unit testing in AngularJS makes this highly difficult to achieve)

- ✓ The minimum coverage should be all happy flow scenarios, followed by validation and then edge cases

- ✓ Unit testing aims to test code units in ***isolation***, using mocks and stubs when necessary

Karma and Jasmine

- ✓ npm install karma karma-jasmine jasmine-core karma-chrome-launcher --save-dev
- ✓ npm install angular angular-ui-router angular-mocks --save-dev
- ✓ npm install -g karma-cli
- ✓ karma init

Karma and Jasmine

```
module.exports = function(config) {  config.set({
    basePath: '',
    frameworks: ['jasmine'],
    files: [],
    exclude: [],
    preprocessors: {},
    reporters: ['progress'],
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['Chrome'],
    singleRun: false,
    concurrency: Infinity
  })
}
```

Unit testing structure

```
describe("the description of the test suite", function() {
        var variables;
        beforeEach(function () {
                // definitions
                // injections
                // mocks
                // spy on
        });
        it ("the description of the unit test" , function () {
                // local variables
                //  calls
                // expect
        });
        // or other describe blocks
})
```

Unit testing elements

✓ In AngularJS test are called spec and are usually located in the same folder with the file containging the element they are covering

✓'Describe' is used to encapsulate a test suit

✓ 'Describe' elements can be nested to for suites of tests

```
describe('Calculator ', function() {
// necessary
        it('should add two numbers correctly', function() {});
}
```

✓'it' is used to describe an actual test

✓Each test must contain at least one assertion, for which we use 'expect'

Unit testing Jasmine matchers

- ✓expect(expression)
  - ✓.toBeTruthy()
  - ✓.not.toBeTruthy()
  - ✓.toBeFalsy
  - ✓.not.tobeFalsy()
  - ✓.toMatch()
  - ✓.not.toMatch()
  - ✓.toContain()
  - ✓.not.toContain()
  - ✓.tobeLessThan()
  - ✓.not.toBeLessThan()
  - ✓.toBeGreaterThan()
  - ✓.not.tobeGreaterThen()
  - ✓.toThrowError()
  - ✓.toEqual()
  - ✓.toHaveBeenCalledWith()

Unit testing Jasmine spy on

- ✓spyOn(expression, name)
  - ✓.and.callThrough()
  - ✓.and.returnValue()
  - ✓.and.ToHaveBeenCalled()
  - ✓.and.callFake()
  - ✓.and.throughError()

Testing Controllers

```
angular.module('app', [])
.controller('PasswordController', function PasswordController($scope) {
  $scope.password = '';
  $scope.grade = function() {
    var size = $scope.password.length;
    if (size > 8) {
      $scope.strength = 'strong';
    } else if (size > 3) {
      $scope.strength = 'medium';
    } else {
      $scope.strength = 'weak';
    }
  };
});
```

LEARN stuff IO

Testing Controllers

```
describe('PasswordController', function() {
 beforeEach(module('app'));
 var $controller;
 beforeEach(inject(function(_$controller_){
  // The injector unwraps the underscores (_) from around the parameter names when matching
  $controller = _$controller_;
 }));
 describe('$scope.grade', function() {
  it('sets the strength to "strong" if the password length is >8 chars', function() {
    var $scope = {};
    var controller = $controller('PasswordController', { $scope: $scope });
    $scope.password = 'longerthaneightchars';
    $scope.grade();
    expect($scope.strength).toEqual('strong');
  });
 }); });
```

LEARN stuff IO

Testing Filters

```
myModule.filter('length', function() {
  return function(text) {
    return ('' + (text || '')).length;
  }
});
```

Testing Filters

```
describe('length filter', function() {
  var $filter;
  beforeEach(inject(function(_$filter_){
    $filter = _$filter_;
  }));
  it('returns 0 when given null', function() {
    var length = $filter('length');
    expect(length(null)).toEqual(0);
  });
  it('returns the correct value when given a string of chars', function() {
    var length = $filter('length');
    expect(length('abc')).toEqual(3);
  });
});
```

Testing Directives

```
app.directive('aGreatEye', function () {
    return {
        restrict: 'E',
        replace: true,
        template: '<h1>lidless, wreathed in flame, {{1 + 1}} times</h1>'
    };
});
```

Testing Directives

```
describe('Unit testing great quotes', function() {
  var $compile,$rootScope;
  // Load the myApp module, which contains the directive
  beforeEach(module('myApp'));
beforeEach(inject(function(_$compile_, _$rootScope_){
    $compile = _$compile_;
    $rootScope = _$rootScope_;
  }));
  it('Replaces the element with the appropriate content', function() {
    // Compile a piece of HTML containing the directive
    var element = $compile("<a-great-eye></a-great-eye>")($rootScope);
    // fire all the watches
    $rootScope.$digest();
    // Check that the compiled element contains the templated content
    expect(element.html()).toContain("lidless, wreathed in flame, 2 times");
  }); });
```

Mocking Services

```
angular.module('sampleServices', [])
  .service('util', function() {
    this.isNumber = function(num) {
      return !isNaN(num);
    };

    this.isDate = function(date) {
      return (date instanceof Date);
    };
  });
```

Mocking Services

```
module(function($provide) {
  $provide.service('util', function() {
    this.isNumber = jasmine.createSpy('isNumber').andCallFake(function(num) {
      //a fake implementation
    });
    this.isDate = jasmine.createSpy('isDate').andCallFake(function(num) {
      //a fake implementation
    });
  });
});
//Getting reference of the mocked service
var mockUtilSvc;
beforeEach(inject(function(util) {
  mockUtilSvc = util;
}));
```

Mocking Services

```
angular.module('mockingProviders',[])
  .provider('sample', function() {
    var registeredVals = [];
    this.register = function(val) {
      registeredVals.push(val);
    };
    this.$get = function() {
      function getRegisteredVals() {
        return registeredVals;
      }
      return {
        getRegisteredVals: getRegisteredVals
      };
    };
  });
```

Mocking Providers

```
module(function($provide) {
  $provide.provider('sample', function() {
    this.register = jasmine.createSpy('register');
    this.$get = function() {
      var getRegisteredVals = jasmine.createSpy('getRegisteredVals');
      return {
        getRegisteredVals: getRegisteredVals
      };
    };
  });
});
//Getting reference of the provider
var sampleProviderObj;
beforeEach(inject(sampleProvider) {
  sampleProviderObj = sampleProvider;
}));
```

Mocking Modules

```
angular.module('first', ['second', 'third'])
  //util and storage are defined in second and third respectively
  .service('sampleSvc', function(utilSvc, storageSvc) {
    //Service implementation
  });
```

Mocking Modules

```
beforeEach(function() {
  angular.module('second',[]);
  angular.module('third',[]);
  module('first');
  module(function($provide) {
    $provide.service('utilSvc', function() {
      // Mocking utilSvc
    });
    $provide.service('storageSvc', function() {
      // Mocking storageSvc
    });
  });
});
```

Mocking $q Promises

```
angular.module('moduleUsingPromise', [])
  .factory('dataSvc', function(dataSourceSvc, $q) {
    function getData() {
      var deferred = $q.defer();
      dataSourceSvc.getAllItems().then(function(data) {
        deferred.resolve(data);
      }, function(error) {
        deferred.reject(error);
      });
      return deferred.promise;
    }
    return {
      getData: getData
    };
  });
```

LEARN stuff IO

Mocking $q Promises

```
module(function($provide) {
  $provide.factory('dataSourceSvc', function($q) {
    var getAllItems = jasmine.createSpy('getAllItems').andCallFake(function() {
      var items = [];
      if (passPromise) {
        return $q.when(items);
      }
      else {
        return $q.reject('something went wrong');
      }
    });
    return {
      getAllItems: getAllItems
    };
  });
});
```

Mocking $q Promises

```
it('should resolve promise', function() {
  passPromise = true;
  var items;
  dataSvcObj.getData().then(function(data) {
    items=data;
  });
  rootScope.$digest();
  expect(mockDataSourceSvc.getAllItems).toHaveBeenCalled();
  expect(items).toEqual([]);
});
```

Mocking Global Variables

```
angular.module('someModule').service('storageSvc', function($window) {
  this.storeValue = function(key, value) {
    $window.localStorage.setItem(key, value);
  };
});
angular.module('globalObjects',[]).constant('toastr', toastr);
```

Mocking Global Variables

```
beforeEach(function() {
  module(function($provide) {
    $provide.constant('toastr', {
      warning: jasmine.createSpy('warning'),
      error: jasmine.createSpy('error')
    });
  });
  inject(function($window) {
    window = $window;
    spyOn(window.localStorage, 'getItem');
    spyOn(window.localStorage, 'setItem');
  });
});
```

THANK YOU!

Vlad Costel Ungureanu
ungureanu_vlad_costel@yahoo.com